Chameleon Information Management Services Limited

INFOFLEX v5
QUERY DESIGN MANAGER
USER GUIDE

© Chameleon Information Management Services Ltd 2015. All rights reserved.

No reproduction, copy or transmission of this publication or any part of or excerpt therefrom may be made in any form or by any means (including but not limited to photocopying, recording, storing in any medium or retrieval system by electronic means whether or not incidentally to some other use of this publication or transiently) without the written permission of Chameleon Information Management Services Limited or in accordance with the provisions of the Copyright Designs and Patents Act 1994 (as amended). Any person who does an unauthorised act in relation to this copyright work may be liable to criminal prosecution and/or civil claims for damages.

Document control

Document name	Quercy Design Manager User Guide
Confidentiality	Customer use
Owner	Jenny Wattis
Version	1.4
Last revised by	JW
InfoFlex version	5.60.0100
Last revised date	July 2015
Status	Customer

Document history

Date	Doc version	Ifx version	Editor	Change
September 2010	1.1		JW	New document
September 2010	1.1b		JW	Proof reading
August 2012	1.2	5.50.0200	JW	Updates for 5.50.0200. Updates to operators list 4.2.2 and filtering with memo items 4.7.7. Launch QDM from toolbar. 2.1.2 index column in view. Default setting of Outer Join property 5.2.5, 7.4 5.3 prompt to save before running a query.
January 2014	1.3	5.60.0100	JW	Updates for 5.60.0100 New Median function in query view Information functions as default values for prompts
July 2015	1.4	5.60.0100	JW	Update to 4.2.2 to clarify <> and Not IN operators don't return null values.

CONTENTS

About this document **5**

1 About Query DesignManager **6**
1.1 About Query Design Manager 6
1.2 Domain and data view queries 6
1.3 Accessing QDM 7
1.4 Query Groups 9
1.4.1 Query groups in a domain 9
1.4.2 Query groups in a data view 10
1.4.3 Viewing query groups belonging to other users 11
1.5 Defining query groups 13
1.6 Exercise 14

2 About queries **15**
2.1 Navigation 16
2.1.1 Reviewing query definitions 16
2.1.2 Reviewing view definitions 17
2.1.3 Reviewing filter definitions 18
2.2 Exercise 19

3 Defining views **20**
3.1 Creating a view 20
3.2 Adding items to the view 21
3.2.1 Re-ordering items 22
3.3 View Item Properties 23
3.3.1 Formats 23
3.3.2 Options 23
3.3.3 Sort 23
3.3.4 Group by 23
3.3.5 Hide 23
3.3.6 Alias 23
3.4 Exercise 24

4 Defining filters **25**
4.1 Creating a filter 25
4.2 Filter criteria 26
4.2.1 Adding items to the filter 26
4.2.2 Operators 28
4.2.3 Entering values 29
4.3 Filters using multiple criteria 30
4.4 Filters using a mixture of ANDs and ORs 31
4.5 Re-ordering within filters 32
4.6 Deleting items from filters 32
4.7 Examples of some filter criteria 33
4.7.1 Filters using IN 33
4.7.2 Filters using CONTAINS CODE 33
4.7.3 Filters using LIKE 34
4.7.4 Filters using NULL 34
4.7.5 Filters using KNOWN 35
4.7.6 Filters using MISSING 35
4.7.7 Filtering with memo items 36
4.8 Exercise 37

5 Defining a query **38**
5.1 Creating a query 38
5.2 Adding properties to a query 39
5.2.1 Description 39
5.2.2 View 39
5.2.3 Filter 39
5.2.4 Link at 40
5.2.5 Outer Join 40
5.2.6 Distinct Rows 40
5.2.7 Context 40
5.2.8 User-defined 40

5.3 Running a query 41
5.3.1 Studies and queries 42
5.3.2 Testing query results 42
5.3.3 Viewing SQL 43
5.4 Event counting 44
5.5 Exercise 45

6 Prompt Filters 46
6.1 Defining a prompt filter 46
Running a query containing a prompt filter 47
6.2 Exercise 48

7 Query Parameters 49
7.1 Types of query 49
7.2 About Linking 49
7.2.1 How to set the Link Level 51
7.2.2 Exercise 52
7.3 Distinct Rows 53
7.3.1 How to set Distinct Rows 54
7.3.2 Exercise 54
7.4 Joining 55
7.4.1 How to set the Join 56
7.4.2 Exercise 57
7.5 Context 58
7.6 User-defined 58

8 View parameters – Occurrence Counting 59
8.1 Occurrence Counting 59
8.2 Exercise 61

9 Manipulating Data in Views 62
9.1 Functions 62
9.1.1 Count() 62
9.1.2 Count(Distinct()) 63
9.1.3 Min() 63
9.1.4 Max() 63
9.1.5 Sum 64
9.1.6 Avg() 64
9.1.7 StDev() 64
9.1.8 Median() 65
9.1.9 Abs() 65
9.1.10 Upper(), Lower() 66
9.1.11 Ltrim, Rtrim 66
9.2 Multiple functions 68
9.3 Aggregated values grouped by patient 68
9.4 Concatenation 69
9.5 Fixed Values in Views 70
9.6 Expressions using the Formula Builder 71
9.6.1 Expressions without the formula builder 78
9.7 Simple Calculations in Views 79
9.7.1 How to define calculations in views 79
9.8 Summary of Operators that can be added to views 81
9.9 Exercises 82
9.9.1 Functions and multiple functions 82
9.9.2 Aggregated values grouped by patient 82
9.9.3 Concatenation 83
9.9.4 Fixed values 83
9.9.5 Expressions in views 83
9.9.6 Simple calculations 83

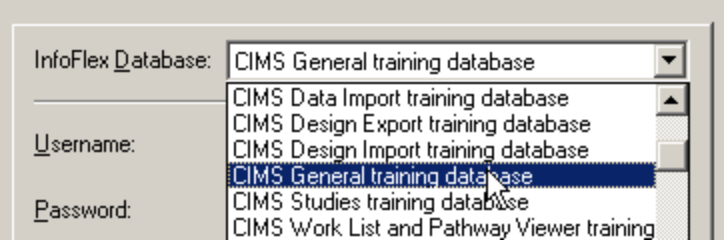
10 Comparisons and Calculations in Filters 84
10.1 Comparison of Fields 84
10.2 Calculations in Filters 85
10.2.1 Adding functions to filters 86
10.2.2 Add Prompt 86
10.3 Calculations in Filters using Fields containing Blanks 88
10.4 Summary of Operators that can be added to filters 90
10.5 Using Subfilters in Filters 91
10.6 Using Subqueries in Filters 93

10.6.1 Selecting a subquery in a filter 94
10.7 Exercises 95
10.7.1 Comparison of fields 95
10.7.2 Calculations in filters 95
10.7.3 Subfilters 96
10.7.4 Subqueries 96
11 Exporting data 97
11.1 Export to Microsoft Excel 97
11.2 Export to File 99
12 Moving and copying views, filters and queries 102
13 Syntax Differences between SQL and Access 104
13.1 Interval arguments in date functions 104
13.2 Date calculations 104
13.3 Functions in filters 105
13.4 Views and filters 105
13.4.1 IFNULL, IFMISSING, IFUNKNOWN, IFMISSINGORUNKNOWN 105
13.4.2 IsNull 105
13.4.3 IIF 105
13.4.4 Count(Distinct) 105
13.5 String concatenation 105

About this document

This document is a reference guide for the InfoFlex Query Design Manager module.
This document can also be used as a training guide in conjunction with the CIMS General training database. Wherever appropriate, exercises are included at the end of a section.

The exercises in this document use the **CIMS General training database.**



The Username is **training** and the Password is **training**.

Before starting the exercises, you should login to the **CIMS General training database**, go to Design Management and ensure that the following domains and data views are unarchived:

- Clinical Domain
- Clinical Data view
- Training Domain
- Training Data view

This document assumes that the user is familiar with InfoFlex Design Management 1 and 2.

1 ABOUT QUERY DESIGNMANAGER

1.1 About Query Design Manager

Query Design Manager is the InfoFlex tool that allows the user to define views, filters and queries and to run queries to extract data.

A View defines which items of data will be returned.
A Filter defines which subset of patients or records you wish to view the data for.
A Query links a view and a filter together and sets certain query parameters which control how the view and filter are linked together.

In Query Design Manager you can view and edit existing views, filters and queries and you can create new views, filters and queries. You can also run queries and export the resulting data.

Queries are used in many places in InfoFlex eg

- Data Analysis
- Reporting
- Work List
- Data Entry subject searches
- Scheduler
- Bed Manager
- Add-Ins eg the Extract Add-In for the production of data extracts.

Within QDM, the following symbols are used:

represents a query group

represents a query

represents a view

represents a filter

Query Design Manager will be referred to as **QDM** throughout this manual.

1.2 Domain and data view queries

Queries can be defined both within domains and within data views. Queries cannot be moved or copied between a domain and a data view, so it is important to define your queries in the correct location. The location a query should be created in is governed by the purpose of the query.

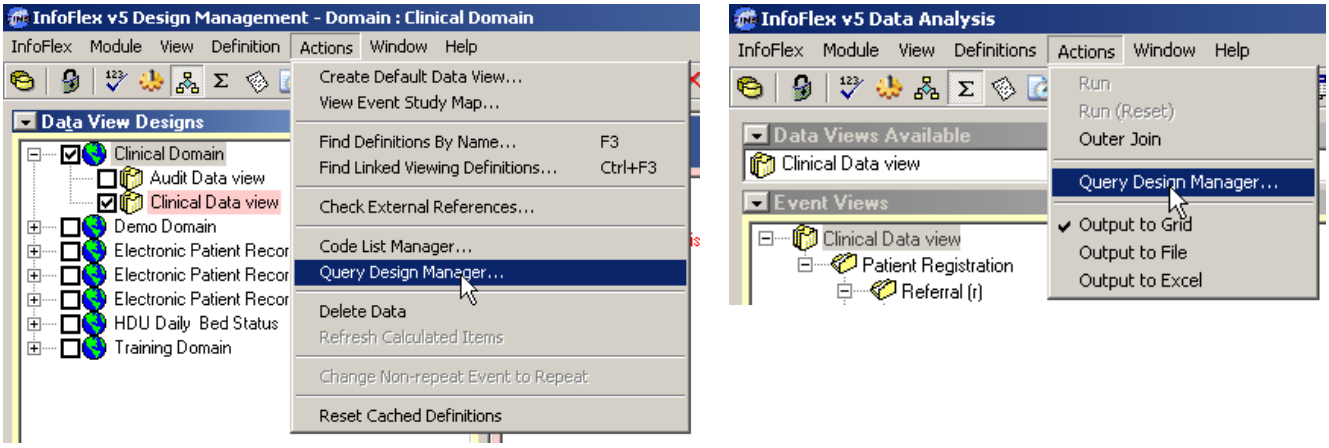
For example, queries for use in documents and reports are defined on the domain since documents and reports are defined at domain level. However, queries for use in Data Analysis are defined in the Data View since access to Data Analysis is granted by data view, and the data items available need to be limited by user permission.

When defining a query for a particular purpose, it is therefore important to know whether the query should be in a domain or a data view before defining it. Note that event view summaries can use both domain and data view queries. Queries defined on the domain can be used in event view summaries in any data view, whereas queries defined on a data view are only available for event view summaries defined within that data view.

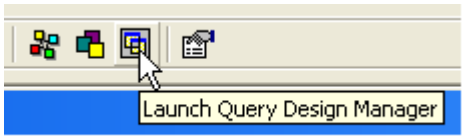
The examples shown in this document use queries in **data views**, however the functionality is the same whether queries are being defined in domains or data views.

1.3 Accessing QDM

QDM can be accessed from the **Actions** menu in the Design Management and Data Analysis modules.



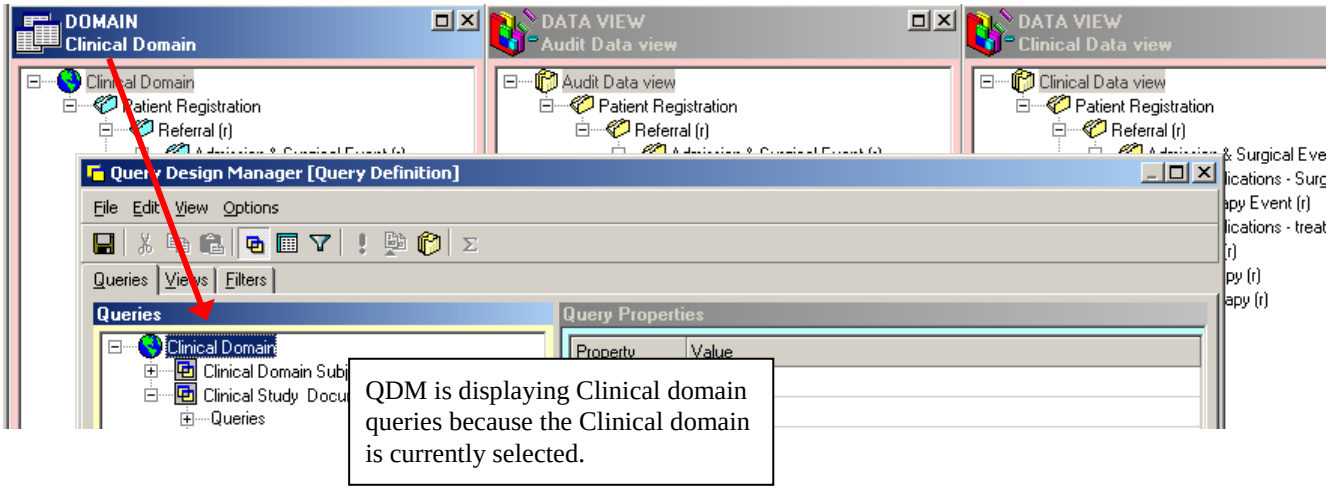
QDM can also be launched from toolbar buttons in the Design Management and Data Analysis modules.

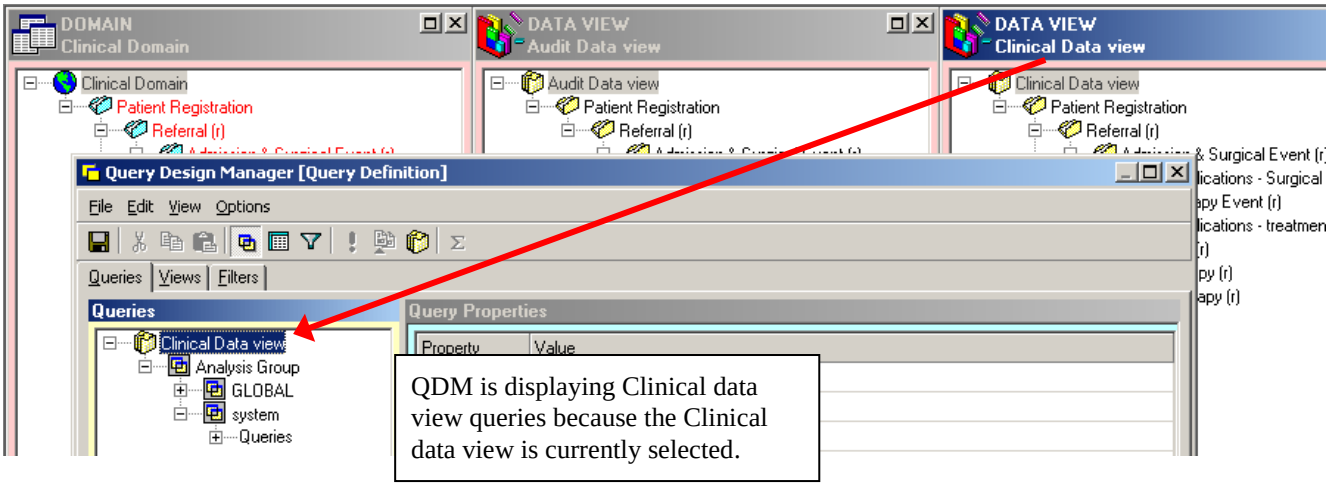


QDM displays one domain or one data view at a time. You must select the domain or data view in which you want to define your queries **before** you open QDM.

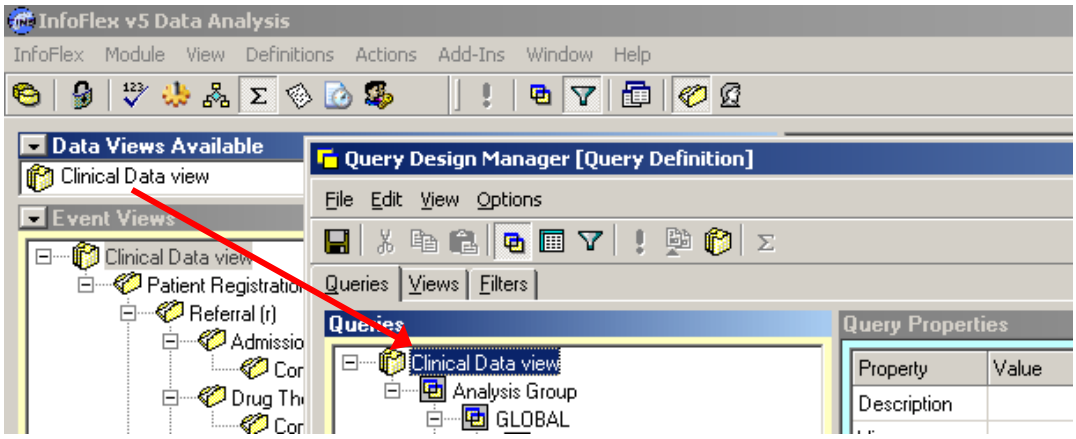
When QDM is opened, it displays either domain queries or data view queries depending on the context from which it has been opened.

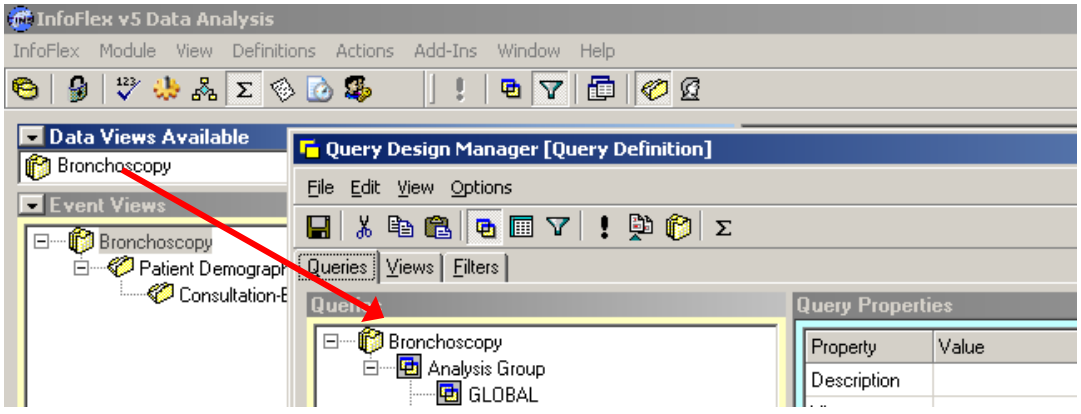
When opening QDM from the Design Management module, QDM displays the domain or data view which is currently selected.





When opening QDM from the Data Analysis module, QDM displays the data view which is currently selected for analysis.





1.4 Query Groups

Query Groups allow the grouping of Views, Filters and Queries to enable them to be managed more easily if a large number have been created. Query groups behave like folders or directories. Some default query groups are defined automatically. Additional query groups can be defined within the default query groups.

The symbol indicates a query group.

The symbol indicates a query.

1.4.1 Query groups in a domain

Within a Domain, two query groups are created by default for **Subject Search** queries and for **Document** queries.

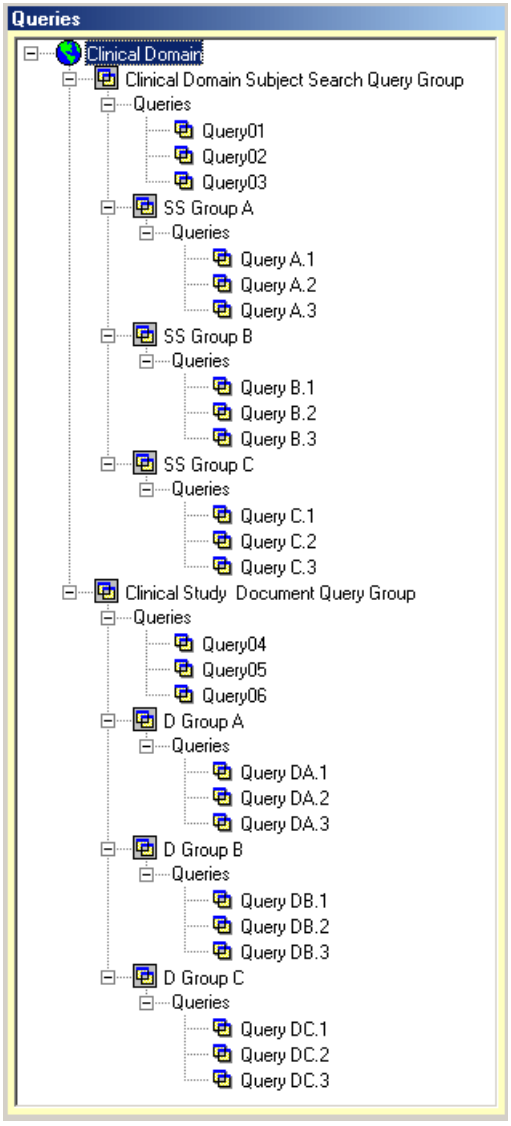
Queries can be copied and pasted between these two groups, however the two groups have their own purposes. Queries should therefore be defined within the appropriate group or they may not be available for selection.

Subject Search queries are for use in Data Entry. They are defined in QDM then selected in a data view definition in Design Management. When selecting subject search queries within a data view definition, only queries within the Subject Search group are available for selection.

Document queries are for use in document and report definitions. When selecting queries within a document or report definition, only queries within the Document group are available for selection.

Query subgroups can be created within each of the above groups to enable management of the queries.

It is recommended that an appropriate structure of query groups and a naming convention for queries and query groups are used.



1.4.2 Query groups in a data view

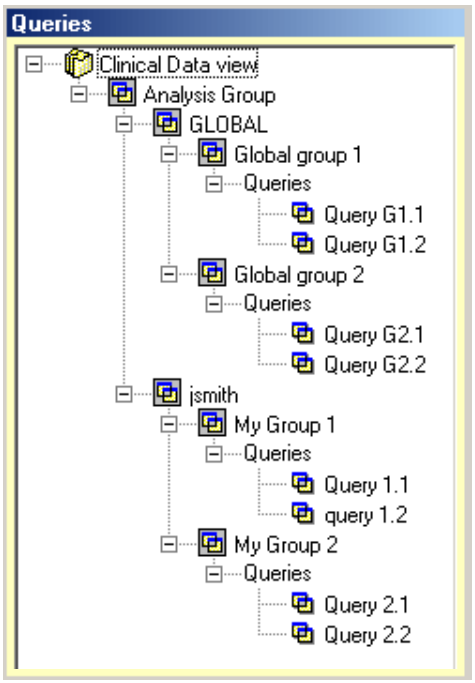
Within each Data View, a query group for Data Analysis queries is created by default.

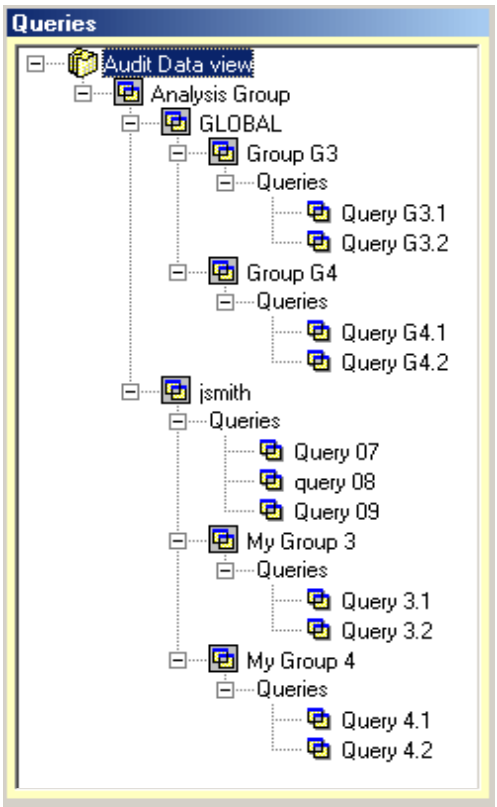
Within this **Analysis Group**, there are further default query groups - a **Global** query group and a query group for each username. Each user can see the **Global** group and the group corresponding to their own username. This is true whether you are accessing data views in QDM from Design Management, Data Analysis or any other function. Administrators have the additional option to view all users' named query groups.

Queries can be copied between groups within the Analysis group (and the Administrator can copy queries between the named users' groups) but queries cannot be copied between data views.

Query subgroups can be created within each of these groups to enable management of the queries.

Queries created within the Analysis group are available whenever a function requires queries to be selected from a data view.



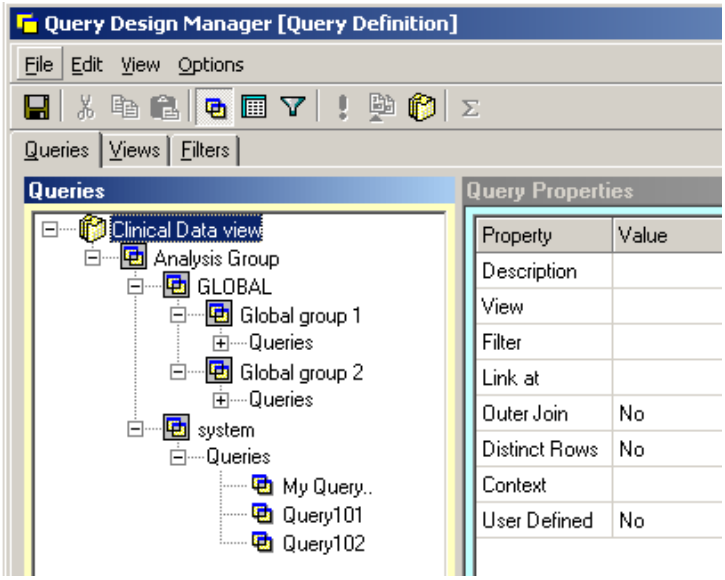


1.4.3 Viewing query groups belonging to other users

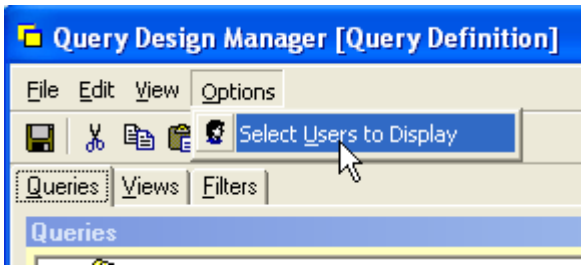
By default, within a data view each user can view their own named query group in addition to the Global query group.

In addition, Administrators can choose to view query groups belonging to other users.

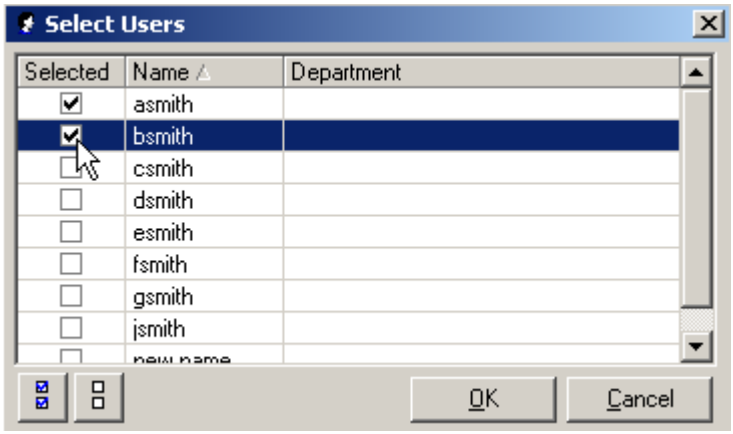
To view another user's query group, open QDM for the appropriate data view.



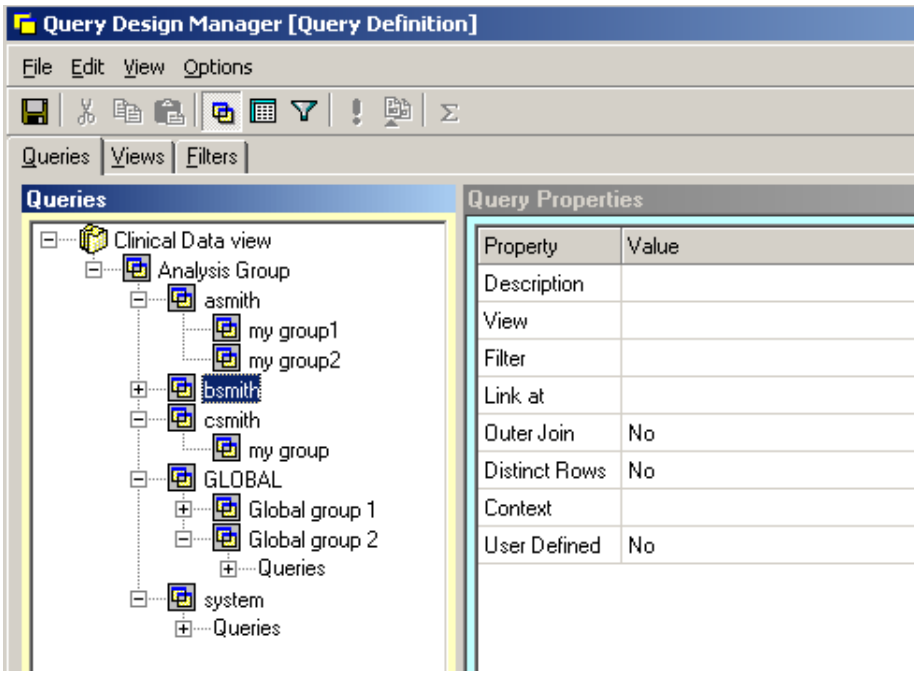
Go to the **Options** menu and choose **Select Users to Display**.



A list of users is displayed. Select which users' groups you wish to view.

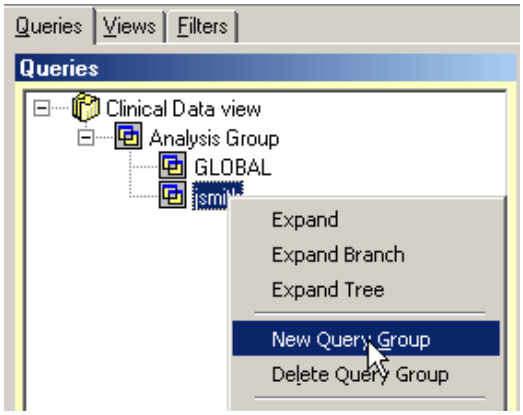


After pressing OK, the selected users' groups are displayed in QDM.

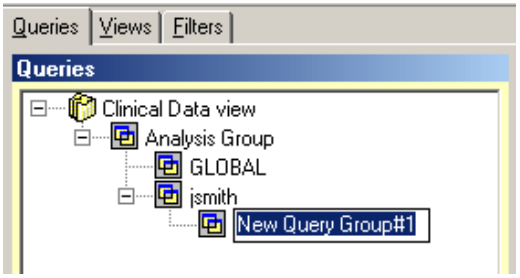


1.5 Defining query groups

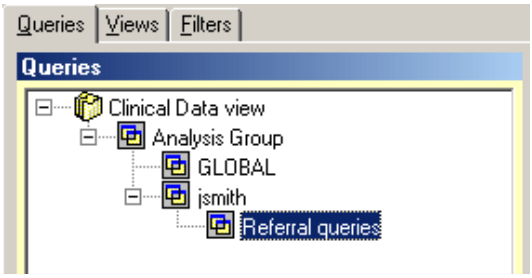
To define a query group, right click the parent group you wish to attach it to and choose **New Query Group**.



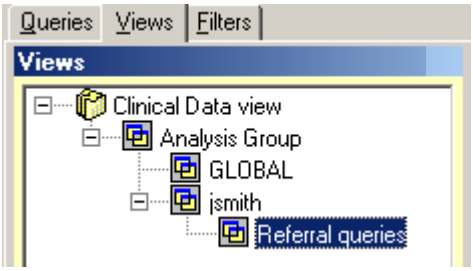
A new query group is displayed on the tree.

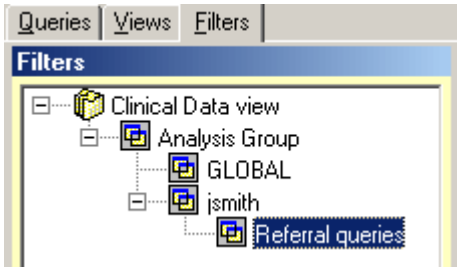


Give the query group a name that describes the queries that will be defined within it and then press return.



The query group is displayed on each of the **Queries**, **Views** and **Filters** tabs.





1.6 Exercise

Log into the InfoFlex **CIMS General training database** using the username **training** and the password **training**.

Go to Design Management and display the **Clinical** domain and the **Clinical** data view. Ensure that both are unarchived.

Select the Clinical Domain and then go to the **Actions** menu and choose **Query Design Manager**. Note that QDM displays the Clinical Domain.

Note that the **Document** query group and **Subject Search** query group are attached to the domain. Some other query groups have been defined within these groups.

Expand the **Queries** heading attached to the Subject Search query group. Note the difference between the symbol for a query and the symbol for a query group.

Close QDM.

In Design Management, select the **Clinical data view**, then go to the **Actions** menu and choose **Query Design Manager**. Note that QDM displays the Clinical Data view.

Note that the **Analysis** query group is attached to the data view. The **Analysis** query group contains the **Global** group and the **training** group.

Create two query groups within the **training** query group. Call them **Training group 1** and **Training group 2**.

2 ABOUT QUERIES

A query combines a **view** and a **filter**.

Views define which items of data will be displayed for the chosen group of patients. A view can contain multiple data items from multiple events. Functions can be applied to items in the view (for example to show the maximum, minimum or average of an item), and calculations can be carried out within a view (for example to add two values together).

Filters define the subset of data that is to be returned, i.e. which group of patients you wish to view (for example, all male patients, or all patients with a certain referral date or a certain consultant). Filter criteria can be defined from any event in the design, and multiple criteria can be defined within one filter.

You can also define filters which prompt the user for certain criteria when the query is run eg you can prompt for a consultant name or for a date range. Prompts reduce the need to define multiple queries based on the same data item.

Queries join a particular view and filter together. The resulting query returns a set of data which meets the view and filter criteria.

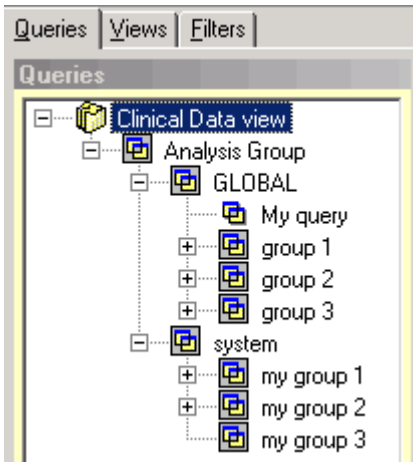
Query parameters can be set to control the behaviour where view items or filter items come from multiple events. **Joining** controls whether a subject can be returned if not all the events represented in the view exist for that subject. **Linking** controls which records are returned when filter criteria come from one or more repeat events. These parameters will be explained later in this document.

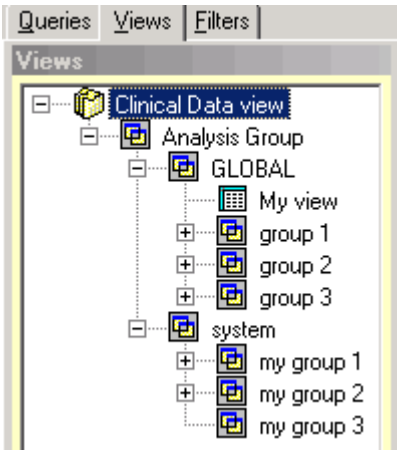
2.1 Navigation

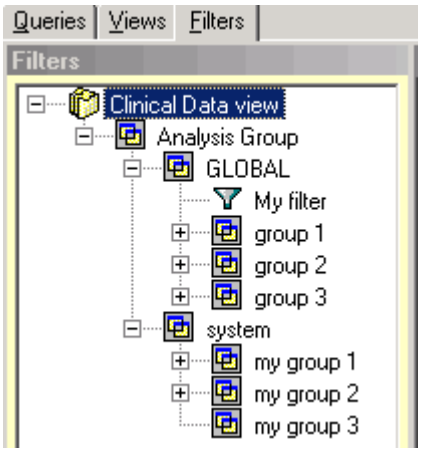
In Query Design Manager there are three tabs:

- the **Queries** tab allows you to view, create and edit queries.
- the **Views** tab allows you to view, create and edit views.
- the **Filters** tab allows you to view, create and edit filters.

Any structure of groups that you create is visible on each of the view, filter and query tabs.



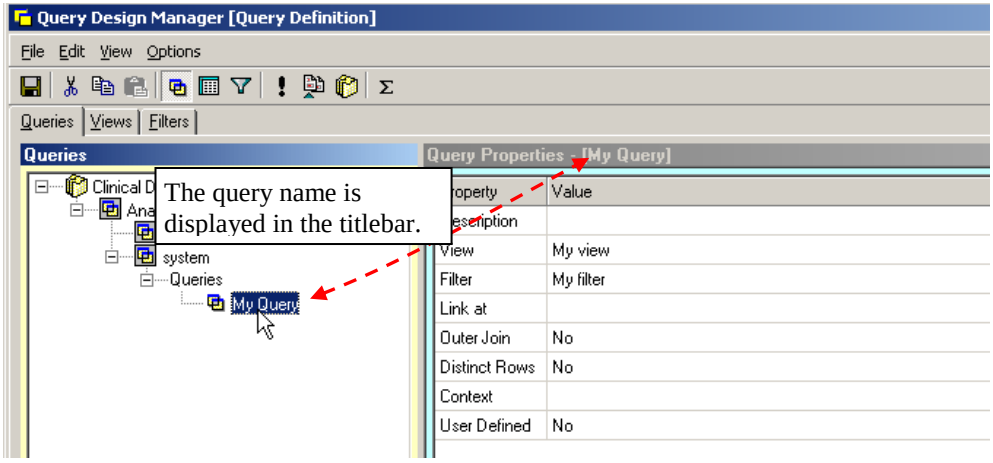




2.1.1 Reviewing query definitions

Whenever you select a definition in the navigation tree on one of the tabs in Query Design Manager, the contents or properties of the definition you have selected are displayed in the main section of the screen, bordered in blue:

When you select a query in the navigation tree on the **Queries** tab, the view, filter and query parameters of the query are displayed. The name of the query is displayed in the titlebar of the main window. You should always double check that the correct name is displayed to ensure that you are viewing the correct query.



If you have selected a query on the **Queries** tab, then when you move to the **Views** tab it will display whichever view is used in the selected query. Similarly if you move to the **Filters** tab, it will display whichever filter is used in that query.

2.1.2 Reviewing view definitions

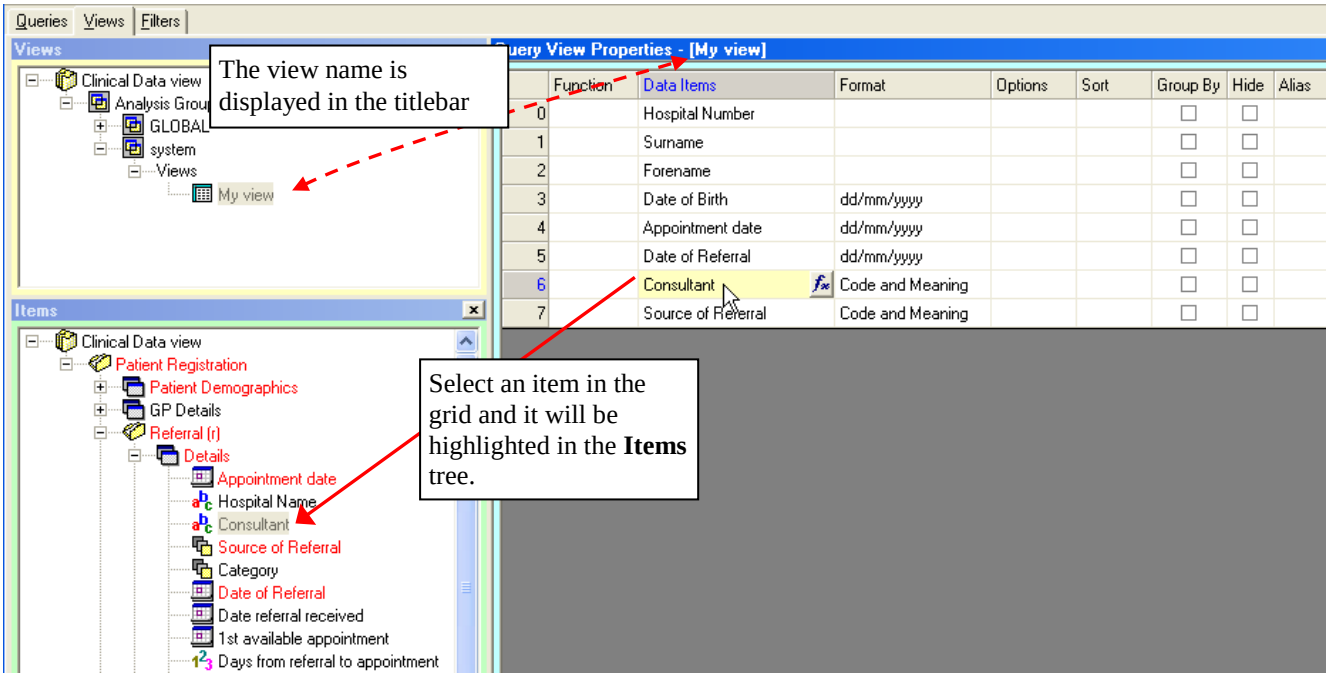
When you select a view on the **Views** tab, the name of the view is displayed in the titlebar of the main window and the items that belong to that view are displayed in a grid beneath the titlebar. (You should always double check that the correct name is displayed in the titlebar to ensure that you are viewing the correct view).

The **Query View Properties** grid shows all the items that exist in the currently selected view. Each row in the view is numbered, starting from zero. (This numbering will be helpful when mapping queries into reports).

The **Items** tree displays all the items in the current domain or data view. Note that when working in a domain, the **Items** tree displays all the events and items in the domain. When working in a data view, the **Items** tree displays the events, panels and items that exist in the data view that you are creating the view in.

If you select a cell in the grid, the row number and column heading of the selected cell are highlighted in blue. Additionally, the **Items** tree is expanded and the item is highlighted in the tree.

Notice that in the **Items** tree, items that already belong to the view are displayed in red, and events (and panels) from which items are selected are also displayed in red.



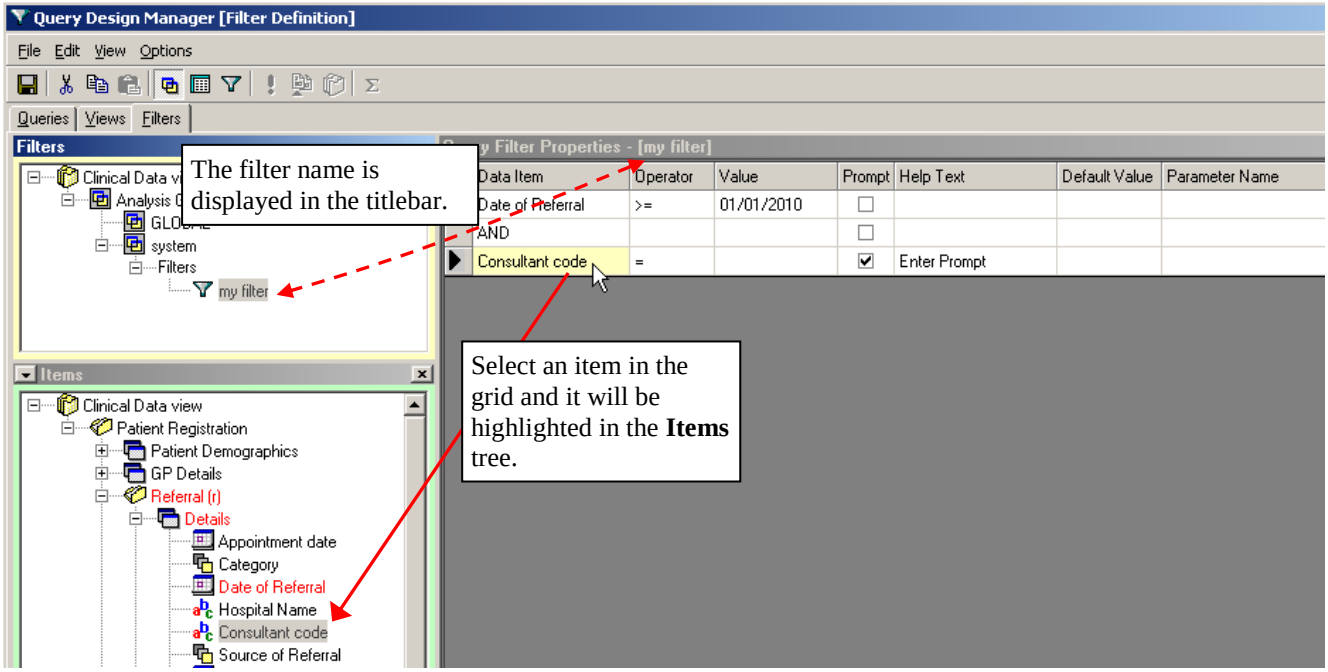
2.1.3 Reviewing filter definitions

When you select a filter on the **Filters** tab, the name of the filter is displayed in the titlebar of the main window and the items that belong to that filter are displayed in a grid beneath the titlebar. (You should always double check that the correct name is displayed in the titlebar to ensure that you are viewing the correct filter).

The **Items** tree displays all the items in the current domain or data view. Note that when working in a domain, the **Items** tree displays all the events and items in the domain. When working in a data view, the **Items** tree displays the events, panels and items that exist in the data view that you are creating the view in.

If you select an item in the grid, the **Items** tree will be expanded and the item will be highlighted in the tree.

Notice that in the **Items** tree, items that already belong to the filter are displayed in red, and events (and panels) from which items have been selected are also displayed in red.



2.2 Exercise

In QDM for the **Clinical** data view, expand the **Queries** heading attached to the **training** query group and select **My query**.

Review the query definition.

Go to the **Views** tab and note the **My view** is selected since it is the view used in the selected query.

Review the view definition.

Go to the **Filters** tab and note the **My filter** is selected since it is the filter used in the selected query.

Review the filter definition.

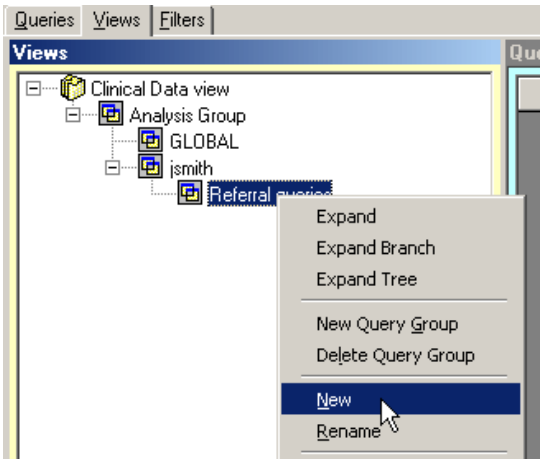
3 DEFINING VIEWS

3.1 Creating a view

When defining views, they must be created within a query group. You must create them in one of the default query groups (if you are in the domain, the Subject Search Query Group or the Document Query Group; if you are in the data view, the Global query group or your username query group).

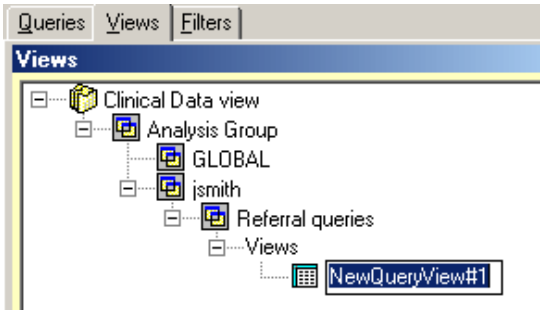
You can create views at the top level within a default query group, however it is recommended that you first define your own structure of query groups within the default query groups. (See section **1.5 Defining query groups**).

To define a view, right click the query group you wish it to belong to, and choose **New**.

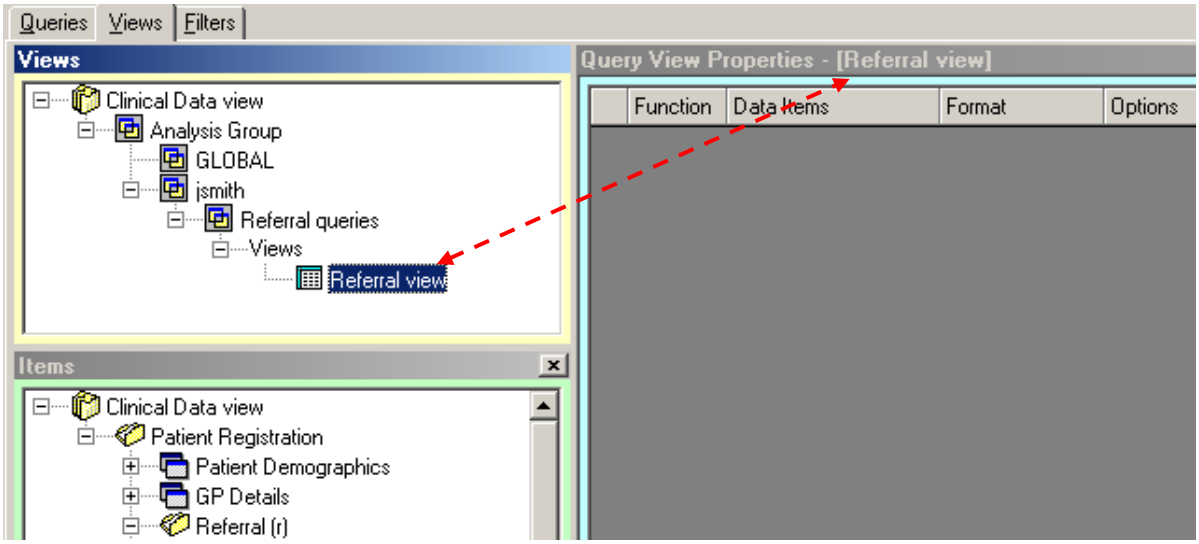


A new query is displayed in the query group.

It is represented by the symbol.



Type a name for your view. It is displayed next to the symbol, and the name is also displayed in the titlebar of the main window. You are now ready to add some items to your view.

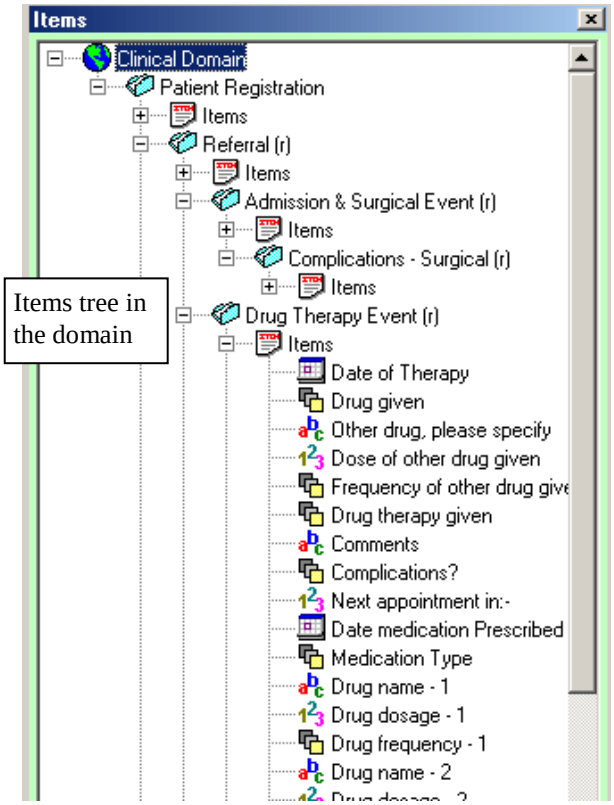


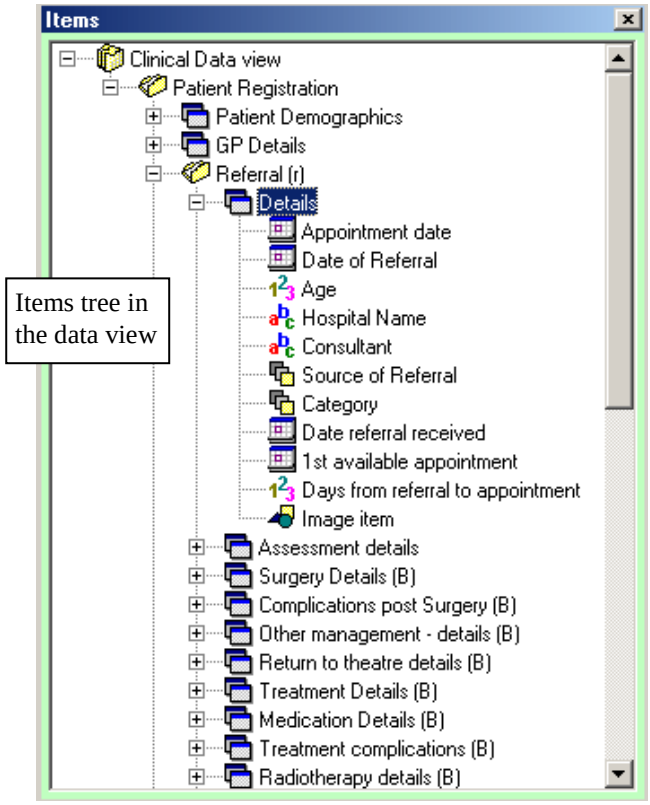
3.2 Adding items to the view

The **Query View Properties** section of the window displays the items that are defined in the view.

To add items to a view, ensure that the correct view is selected, then open the **Items** tree to find the items that you wish to add.

Note that when working in a domain, the **Items** tree displays all the events and items in the domain. When working in a data view, the **Items** tree displays the events, panels and items that exist in the selected data view.





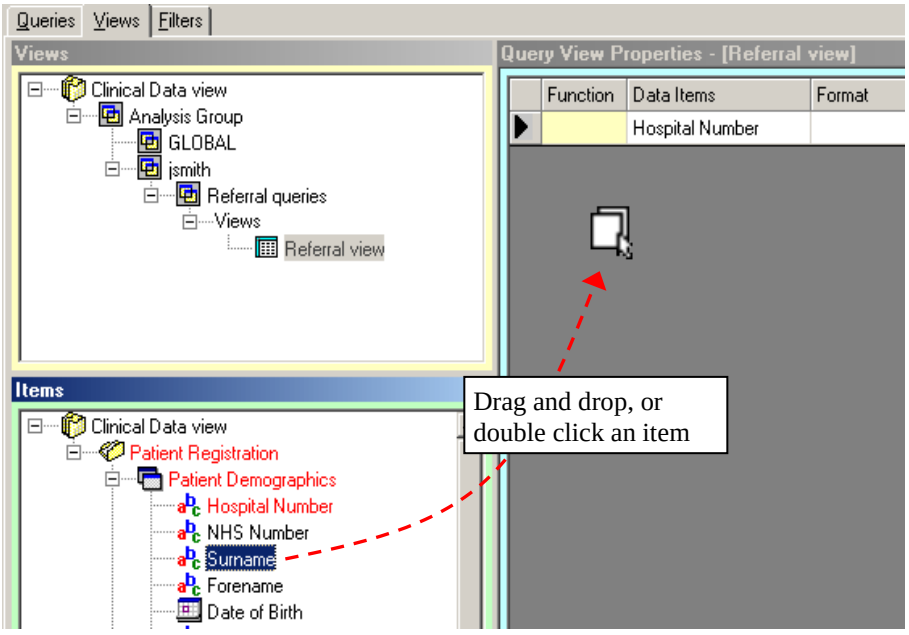
You can add items either by

- dragging them from the **Items** tree into the **Query View Properties** box.

or by

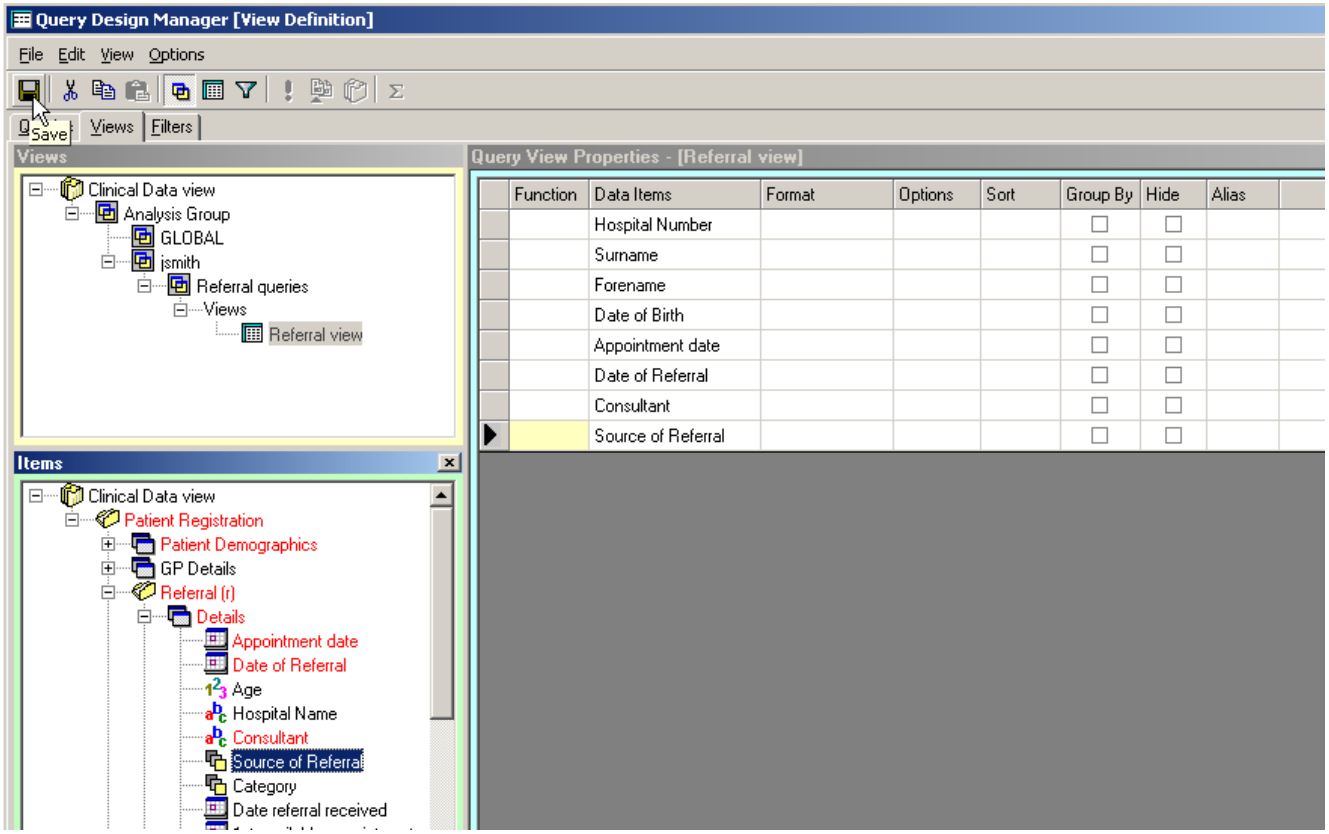
- double clicking the item in the **Items** tree.

Notice that items that have already been selected are displayed in red in the **Items** tree.



You can select as many items from as many different events as required. You can also add the same item twice if necessary. When you have finished, save the view by pressing **F5** or press the **Save** button.

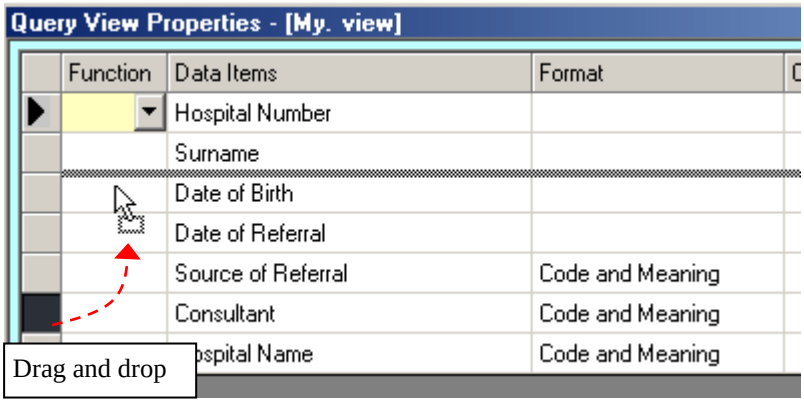
Notice that as you add items to the view, they turn red in the tree, and the events and panels from which items have been selected are selected are also displayed in red.



3.2.1 Re-ordering items

Items can be re-ordered within a view by dragging and dropping.

Pick up the grey cell to the left of the item you wish to move and drag it up or down. Drop it when the grey horizontal line reaches the correct position.

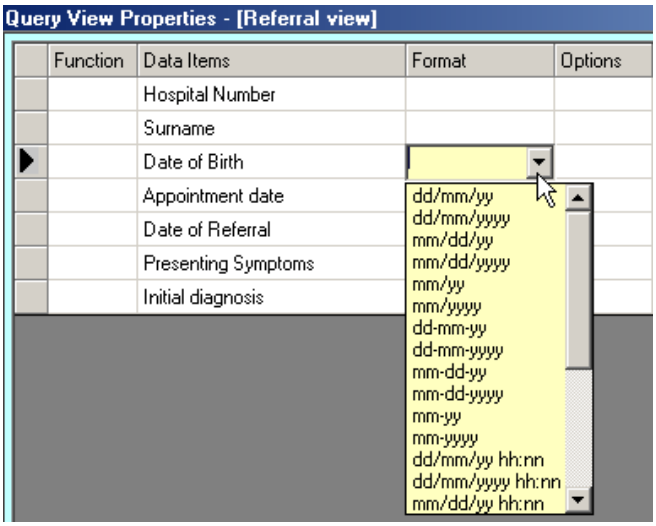


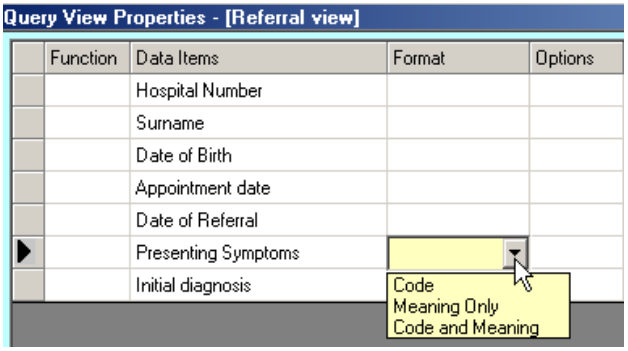
3.3 View Item Properties

3.3.1 Formats

It is possible to specify the format of date, value, coded and dictionary items. These formats over-ride any formats that are set for the items in Design Management.

Formats are set on the **View** tab of **Query Design Manager** in the **Format** column. A dropdown list is displayed where it is possible to set a format.





3.3.2 Options

The **Options** column allows you to specify custom formats for Dictionary Lookup Items that are dates. This is free text data entry but the date format must be a valid date format, eg if you just want to display the month, enter **mmm.**

3.3.3 Sort

The **Sort** option allows you to order the data by a particular column. To use this option, select either **Asc** or **Desc** in the **Sort** column for the item which you wish the data to be sorted by.

3.3.4 Group by

Group by is used in conjunction with the **Count** operator to calculate occurrences of a particular value in an item (see section **8.1 Occurrence Counting**) and also with aggregated values (see section **9.3 Aggregated values grouped by patient**).

3.3.5 Hide

This is a simple tick option and allows you to hide the results from this column. When the query is run, that column of data is simply omitted. This option is particularly useful if you wish to temporarily anonymise data.

3.3.6 Alias

Alias If you wish to specify your own column heading for a particular item, enter it in the **Alias** column. By default no alias is set and the item name is used as the column heading.

3.4 Exercise

In the **Training group 1** group, create a new view called **Referral view**.

Add the following items:

From the **Patient Registration** event:
Hospital number, Surname, Date of Birth

From the **Referral** event, **Details** panel:
Appointment date, Date of referral

From the **Referral** event, **Assessment Details** panel:
Presenting Symptoms, Initial Diagnosis

Set formats for the date items.

Set formats for the dictionary and coded items (choose code & meaning).

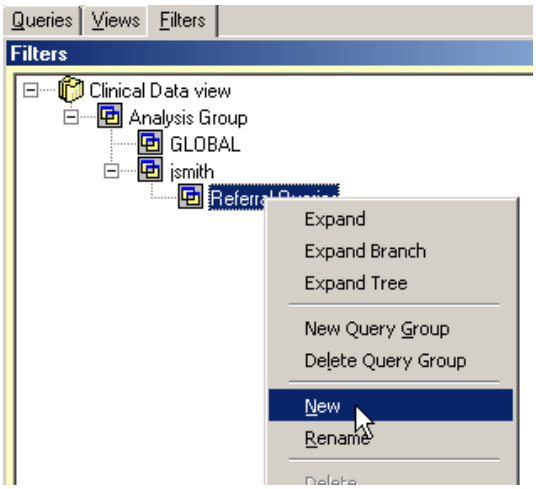
Save the view.

4 DEFINING FILTERS

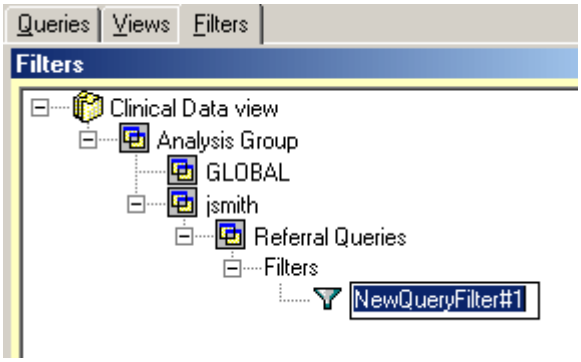
Filters allow you to specify a particular set of patients that you wish to view data for. You can set up a filter that contains only one criteria, (eg all female patients, or all patients for a particular consultant) or you can set up a filter that combines several different criteria (eg all male patients of a particular age that had a particular type of surgery).

4.1 Creating a filter

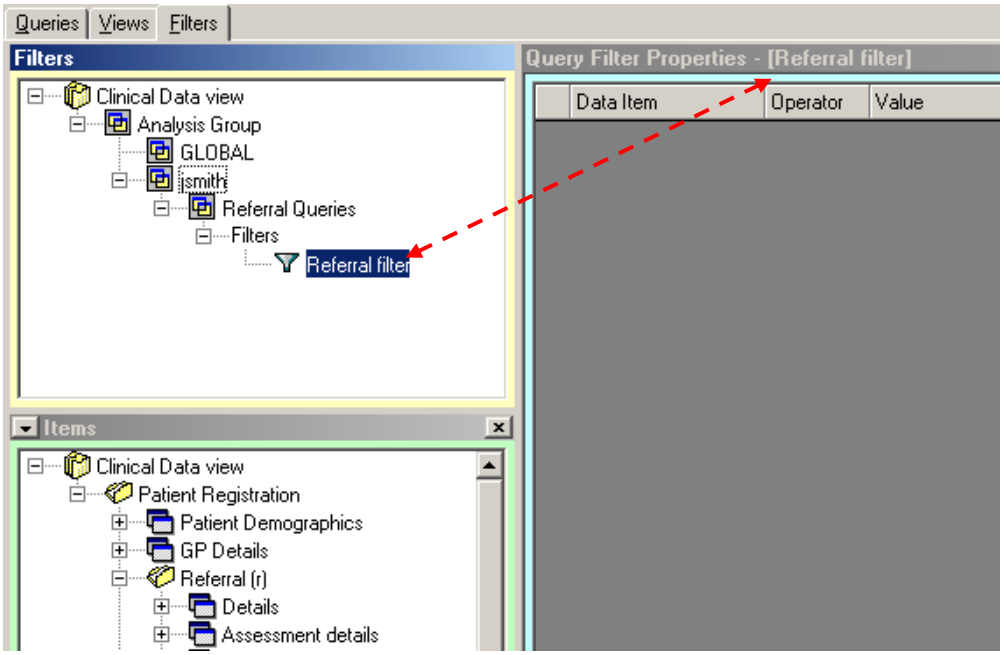
To define a filter, go to the **Filters** tab, right click the query group you wish the filter to belong to, and choose **New**.



A new filter is displayed in the query group. It is represented by the ⧩ symbol.



Type a name for your filter. It is displayed next to the ⧩ symbol, and the name is also displayed in the titlebar of the main window. You are now ready to add criteria to your filter.

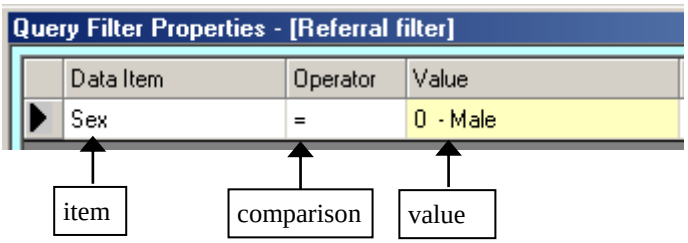


4.2 Filter criteria

The **Query Filter Properties** section of the screen displays the criteria that have been defined in the filter.

Each filter criterion is made up of an **item**, an **operator** and a **value**. The operator specifies how the data item is compared with the value.

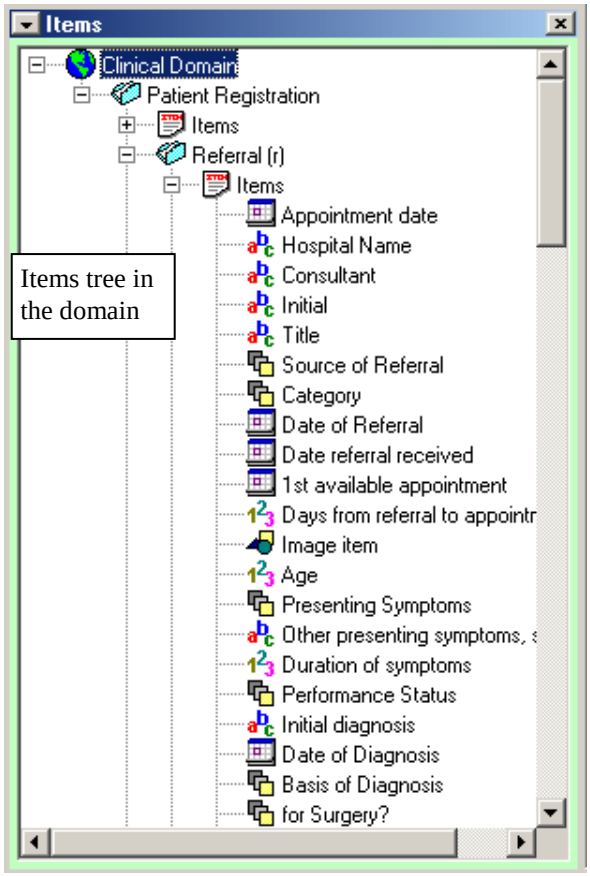
For example a filter criteria to return male patients would be defined as:

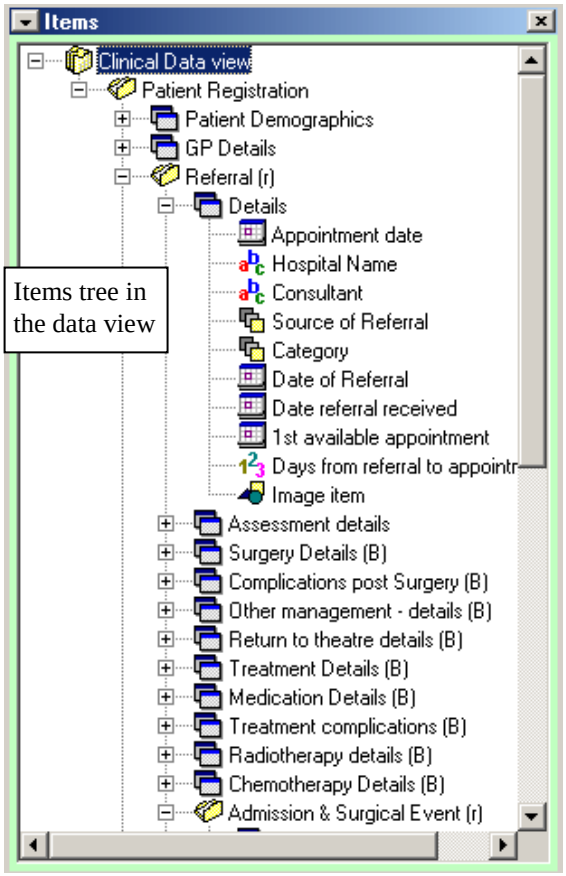


4.2.1 Adding items to the filter

To add an item to the filter, ensure that the correct filter is selected, then open the **Items** tree to find the items that you wish to add.

Note that when working in a domain, the **Items** tree displays all the events and items in the domain. When working in a data view, the **Items** tree displays the events, panels and items that exist in the data view that you are creating the filter in.



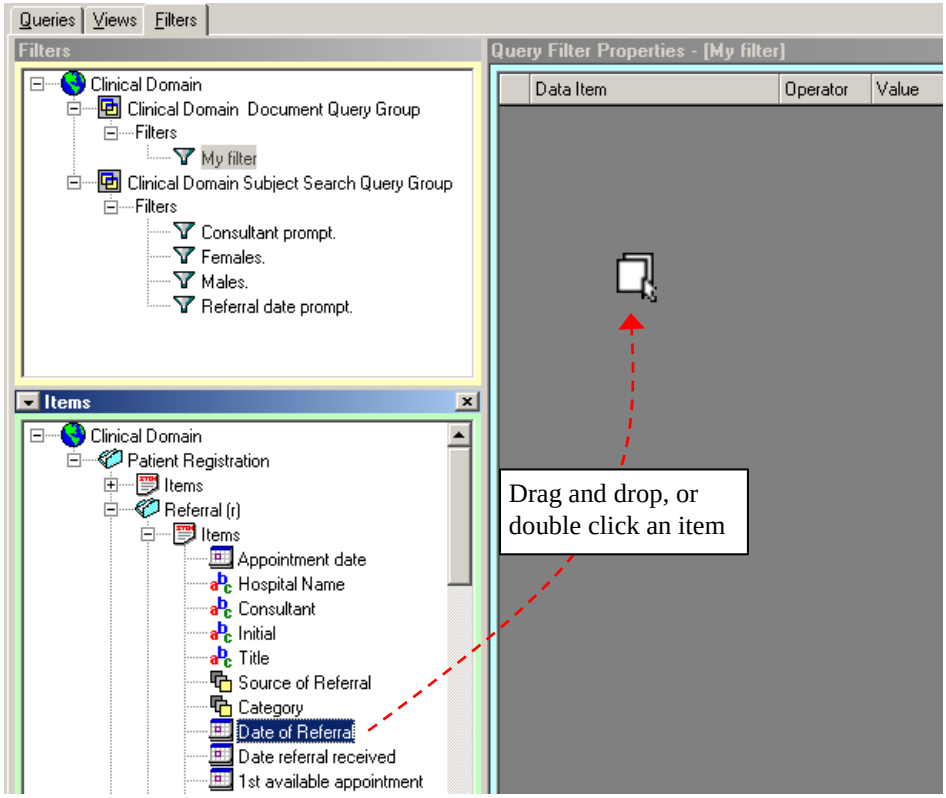


You can add items either by

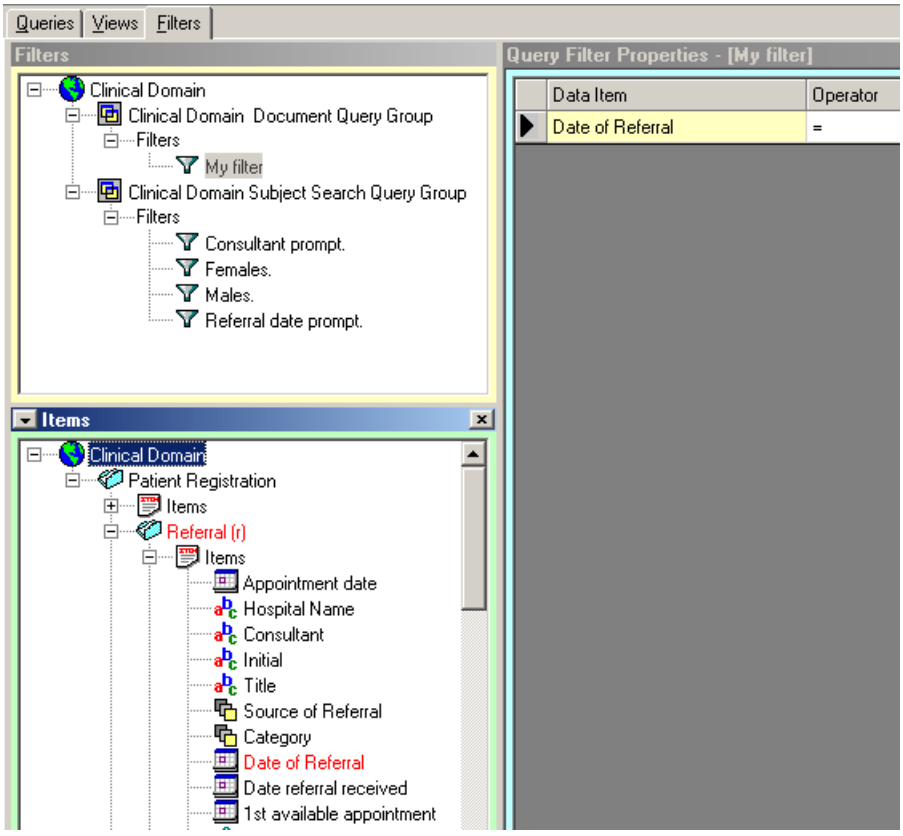
- dragging them from the Items tree into the Query Filter Properties box.

or by

- double clicking the item in the Items tree.



Notice that as you add items to the filter, the items you have added turn red in the tree, and the events and panels from which items have been selected are selected are also displayed in red.



4.2.2 Operators

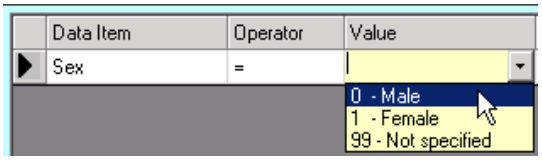
Below is a list of the operators that can be used in filters and their meanings. Some operators are only available for certain item types).

Operator	Meaning	Item types
=	Equals	All types except memo
<>	Not equal to Note that null values will not be returned when using this operator. eg a filter of "DataItem <> 1" will only return records with values entered and will not return records where DataItem has no value entered. An additional criteria of "OR DataItem IS EMPTY" is required.	All types except memo
>	Greater than	All types except memo
>=	Greater than or equal to	All types except memo
<	Less than	All types except memo
<=	Less than or equal to	All types except memo
IN	returns records where the item contains one of the values entered in the Value column	All types except memo
NOT IN	returns records where the item does not contain one of the codes entered in the Value column Note that null values will not be returned when using this operator. eg a filter of "DataItem NOT IN 1,2" will only return records with values entered and will not return records where DataItem has no value entered. An additional criteria of "OR DataItem IS EMPTY" is required.	All types except memo
LIKE	return records where the specified item contains the text entered in the value column. The wildcard * can be used.	All types including memo
NOT LIKE	return records where the specified item does not contain the text entered in the value column	All types including memo
IS EMPTY	returns records where the specified item has no data entered and has not been marked with F11 or F12	All types including memo
IS NOT EMPTY	returns records where the specified item does have data entered or has been marked with F11 or F12	All types including memo
IS NULL	Behaves the same as IS EMPTY. If you select IS NULL, the operator changes to IS EMPTY on saving.	All types except memo
IS NOT NULL	Behaves the same as IS NOT EMPTY. If you select IS NOT NULL, the operator changes to IS NOT EMPTY on saving.	All types except memo
IS KNOWN	returns records where the specified item has data entered, or has been marked with F12, but has not been marked with F11	All types except memo
IS NOT KNOWN	returns records where the specified item has had F11 entered	All types except memo
IS MISSING	The item has been marked with F12	All types except memo
IS NOT MISSING	The item has not been marked with F12 but has data entered, or has been marked with F11	All types except memo
CONTAINS CODE	returns records where the item contains the code specified in the Value column	Multiple Response only

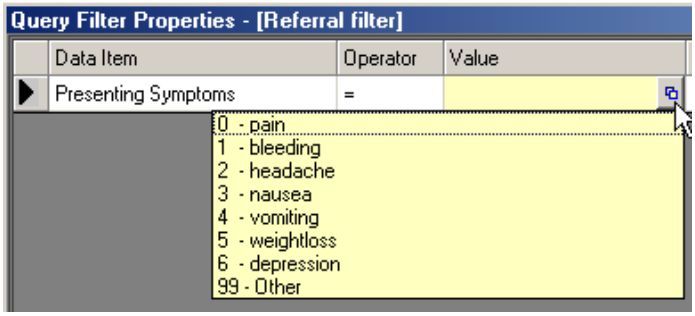
4.2.3 Entering values

When entering values, the value needs to be entered in the same format as in Data Entry. The Value column displays the same selection box that is used in data entry for the item.

For **Coded** items, the **Value** column displays a dropdown list.

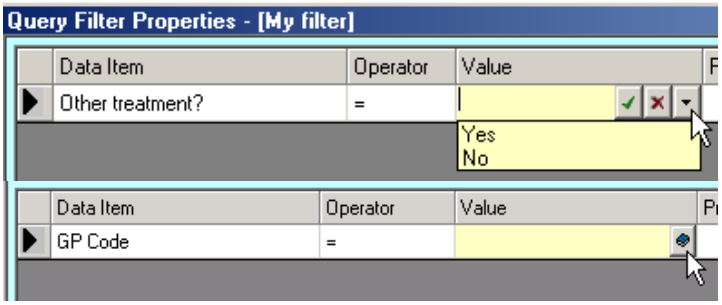


For **MR** items, the **Value** column displays a dropdown list and allows multi-selection.

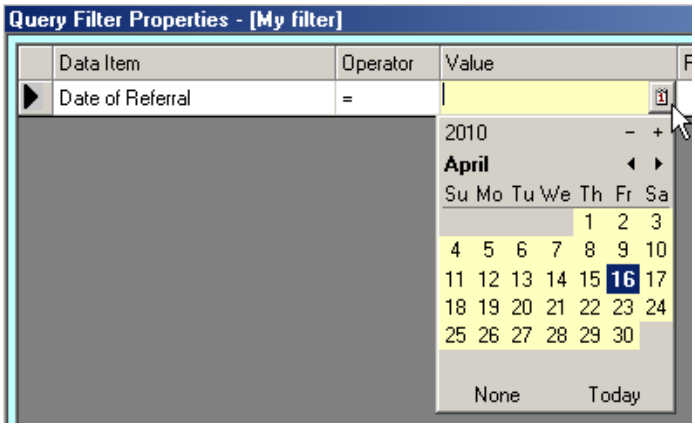


For **Boolean** items, the **Value** column displays the true and false text.

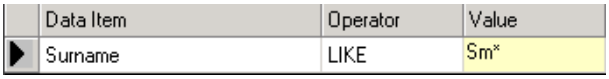
For **Dictionary** items, the **Value** column displays the dictionary search dialog.



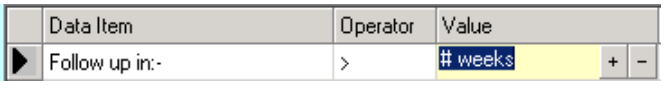
For **Date** items, the **Value** column displays the date picker and enters the date in the format set in the item definition.



For **Text** items, the **Value** column allows free text entry and applies rules regarding case and max length that have been set. The * wildcard can be used.



For **Value** items, the **Value** column displays the format and unit set for the item.



When you have created your filter, save it by pressing **F5** or press the **Save** button.

4.3 Filters using multiple criteria

If you wish to define a filter that uses more than one criterion, the criteria are linked together using **operators**.

The **AND** operator returns records that fulfil both criteria.

The **OR** operator returns records that fulfil either of the criteria.

For example:

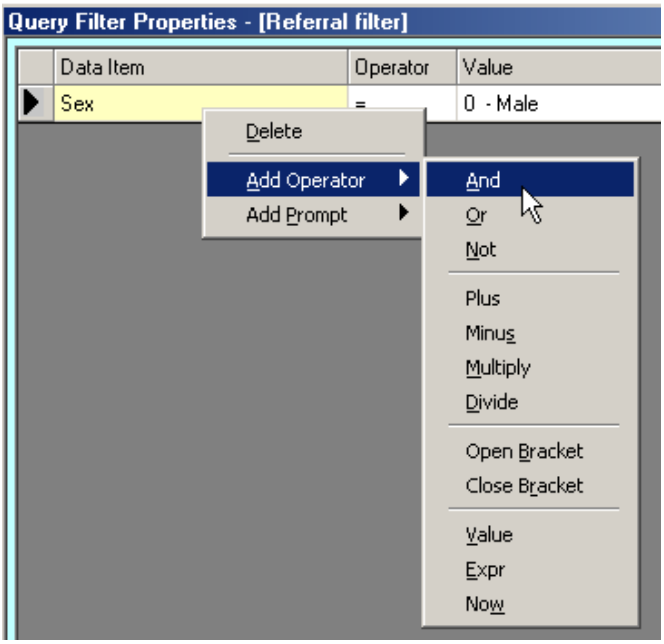
This filter would return only those patients who were both male and single ie each patient has to meet both criteria in order to be returned.

	Data Item	Operator	Value
	Sex	=	0 - Male
	AND		
▶	Marital Status	=	1 - Single

This filter would return all those patients who are male and would also return all those patients who are single ie each patient has to meet only one of the criteria in order to be returned.

	Data Item	Operator	Value
	Sex	=	0 - Male
	OR		
▶	Marital Status	=	1 - Single

To add an **AND** or an **OR**, right click the row beneath which the AND or OR should appear, and choose **Add Operator** then the required operator.



The same item can be used more than once in a filter, if necessary.

4.4 Filters using a mixture of ANDs and ORs

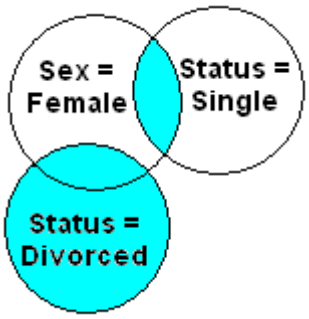
Where a mixture of ANDs and ORs are used in a filter, brackets can be used to define how the criteria are applied. It is possible for the same criteria to return different results with a different arrangement of brackets.

This filter:

Query Filter Properties - [Referral filter]

Data Item	Operator	Value
(		
Sex	=	1 - Female
AND		
Marital Status	=	1 - Single
)		
OR		
Marital Status	=	3 - Divorced

returns this set of patients:

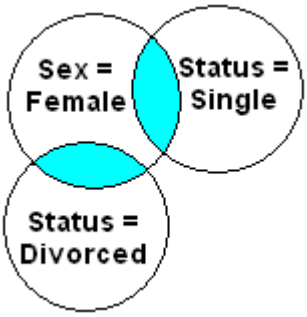


Whereas this filter:

Query Filter Properties - [Referral filter]

Data Item	Operator	Value
Sex	=	1 - Female
AND		
(		
Marital Status	=	1 - Single
OR		
Marital Status	=	3 - Divorced
)		

returns this set of patients:



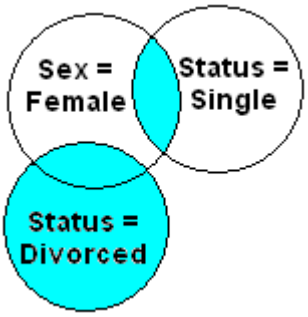
If no brackets were used, then the AND is applied first as in the first example.

This filter applies the AND first:

Query Filter Properties - [Referral filter]

Data Item	Operator	Value
Sex	=	1 - Female
AND		
Marital Status	=	1 - Single
OR		
Marital Status	=	3 - Divorced

and returns this set of patients:

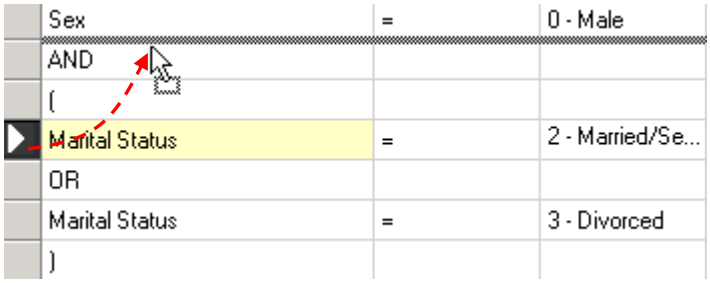


4.5 Re-ordering within filters

When adding items and operators to filters, you can re-order them any time by dragging and dropping.

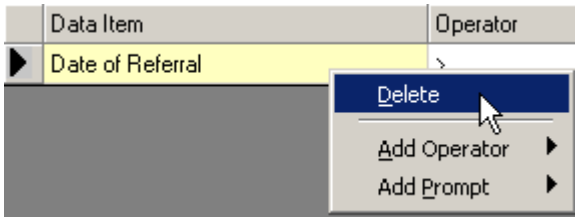
Drag the grey square to the left of any item or operator and move it up or down in the grid.

Drop the item or operator when the grey line is in the correct position.



4.6 Deleting items from filters

To delete any item from a filter, right click the row and select **Delete**.



4.7 Examples of some filter criteria

4.7.1 Filters using IN

The **IN** comparison returns records where an item contains one of several values entered in the Value column. This comparison saves adding multiple criteria based on the same item linked with ORs. Note that when typing in multiple codes, they should be separated by semi-colons.

This filter:

Query Filter Properties - [Referral filter]

	Data Item	Operator	Value
▶	Source of Referral	=	0 - GP
	OR		
	Source of Referral	=	1 - A&E
	OR		
	Source of Referral	=	2 - Dentist

Can be expressed as

	Data Item	Operator	Value
▶	Source of Referral	IN	0 - GP; 1 - A&E; 2 - Dentist

The **NOT IN** comparison will return records where the value stored for the coded item is not one of the codes entered in the Value column. Note though that null values will not be returned when using this operator. eg a filter of "DataItem NOT IN 1,2" will only return records with values entered and will not return records where DataItem has no value entered. An additional criteria of "OR DataItem IS EMPTY" is required.

4.7.2 Filters using CONTAINS CODE

The **CONTAINS CODE** comparison can only be used with multiple response (MR) coded items. It returns records where the data entered in the MR coded item includes the code specified in the Value column. The MR item may contain other codes as well as the one specified.

This filter:

returns all records where **3 – nausea** exists in the **Presenting Symptoms** item

	Data Item	Operator	Value
▶	Presenting Symptoms	CONTAINS CODE	3 - nausea

Presenting Symptoms
0 - pain; 3 - nausea
3 - nausea; 4 - vomiting; 99 - Other
3 - nausea; 5 - weightloss

CONTAINS CODE can be used with AND, OR and NOT to define the filter further, however, brackets are required around the separate CONTAINS CODE clauses, eg:

Query Filter Properties - [Contains1]

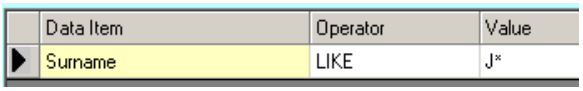
	Data Item	Operator	Value
▶	(		
	Presenting Symptoms	CONTAINS CODE	0 - pain
	)		
	AND		
	NOT		
	(		
	Presenting Symptoms	CONTAINS CODE	3 - nausea
	)		

The example above will include all patients whose **Presenting Symptoms** item contains **0 –pain** but does not contain **3- nausea**. eg It would include **"0;1"** and **"0"**, but not **"0;1;3"**.

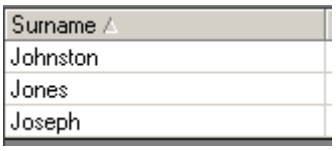
4.7.3 Filters using LIKE

The **LIKE** comparison returns records where the specified item contains the text entered in the value column. This comparison can be used with text items, or to search dictionary codes.
The * wildcard can be used.

This filter returns all records where the surname begins with **J**.



Data Item	Operator	Value
Surname	LIKE	J*

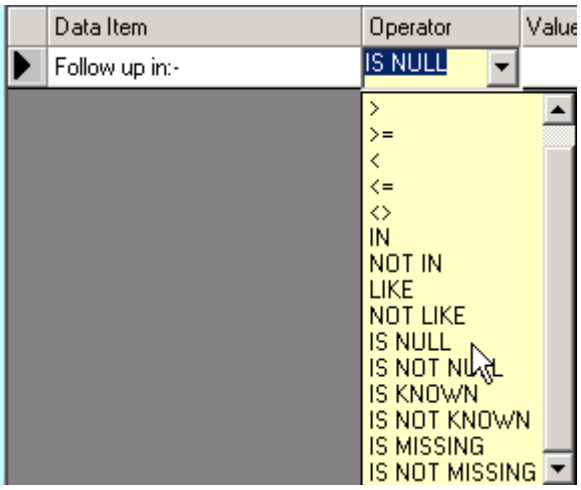


Surname
Johnston
Jones
Joseph

The **NOT LIKE** comparison will return records where the specified item does not contain the text entered in the value column of the filter.

4.7.4 Filters using NULL

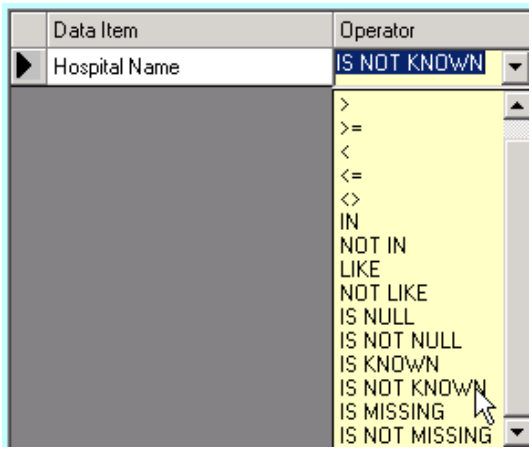
The **IS NULL** comparison will return records where the specified item has no data entered and has not been marked with F11 or F12.



The **IS NOT NULL** comparison will return records where the specified item does have data entered or has been marked with F11 or F12.

4.7.5 Filters using KNOWN

The **IS NOT KNOWN** comparison returns records where the specified item has had F11 entered. F11 puts a green highlight on the item in data entry, and displays –88888 in Data Analysis.



SQL | Results - 3 row(s)

Hospital Number	Surname	Hospital Name
345678	Rose	-88888 - [Unknown]
789012	Parisien	-88888 - [Unknown]
876543	Bartley	-88888 - [Unknown]

The **IS KNOWN** comparison returns records where the specified item has data entered, or has been marked with F12, but has not been marked with F11.

Data Item	Operator
Hospital Name	IS KNOWN

SQL | Results - 11 row(s)

Hospital Number	Surname	Hospital Name
901234	Miles	CD300 - St Luke's Hospital
666666	Smith	BC200 - St Mark's Hospital
123456	Green	AB100 - St Matthew's Hospital
222222	Brown	-99999 - [Missing]
333333	Joseph	DE400 - St John's Hospital
234567	Jones	BC200 - St Mark's Hospital
456789	Gate	-99999 - [Missing]

4.7.6 Filters using MISSING

The **IS MISSING** comparison returns records where the specified item has had F12 entered. F12 puts a blue highlight on the item in data entry, and displays –99999 in Data Analysis.

Data Item	Operator
Hospital Name	IS MISSING

SQL | Results - 3 row(s)

Hospital Number	Surname	Hospital Name
222222	Brown	-99999 - [Missing]
456789	Gate	-99999 - [Missing]
890123	Johnston	-99999 - [Missing]

The **IS NOT MISSING** comparison will return records where the specified item has data entered, or has been marked with F11, but has not been marked with F12.

Data Item	Operator
Hospital Name	IS NOT MISSING

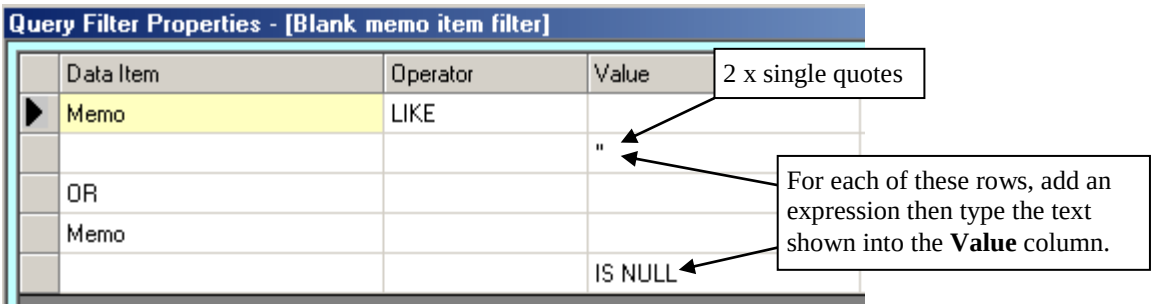
SQL | Results - 11 row(s)

Hospital Number	Surname	Hospital Name
901234	Miles	CD300 - St Luke's Hospital
666666	Smith	BC200 - St Mark's Hospital
123456	Green	AB100 - St Matthew's Hospital
333333	Joseph	DE400 - St John's Hospital
234567	Jones	BC200 - St Mark's Hospital
345678	Rose	-88888 - [Unknown]
567890	Clarke	CD300 - St Luke's Hospital
678901	Mount	DE400 - St John's Hospital
789012	Parisien	-88888 - [Unknown]

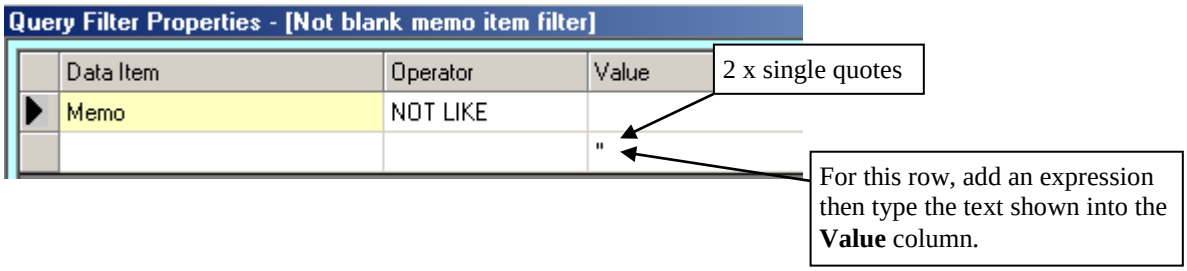
4.7.7 Filtering with memo items

Only the LIKE, NOT LIKE, IS EMPTY and IS NOT EMPTY operators can be used with memo items.

To retrieve memo items that are either empty or null, the following syntax should be used:



To retrieve memo items that are not null or not empty, the following syntax should be used:



4.8 Exercise

1 In the **Training group 1** group, create a new filter called **Referral filter**.

Add the following items:

From the **Referral** event, **Details** panel:
Date of referral

In the Operator column, select >

In the Value column enter **01/01/2000**.

Save the filter.

2 Create a filter to find patients who are both female and have the GP with code X9870004

3 Create a filter to find patients whose surname is Smith or Jones and who are aged over 60 (use the Age item on the Assessment Details panel of the Referral event).

4 Create a filter to find Referrals where the source of referral is 0 – GP, 1 – A&E, or 2 – Dentist.

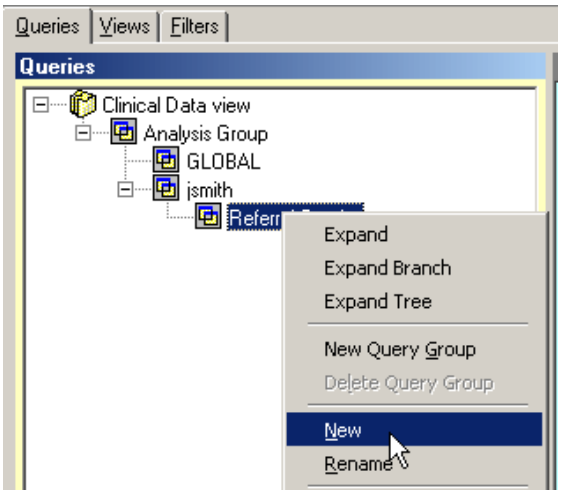
5 Create a filter to find Referrals where the patient's Presenting Symptoms include 3 – nausea.

6 Create a filter to find Referrals where the Source of Referral is marked as Missing.

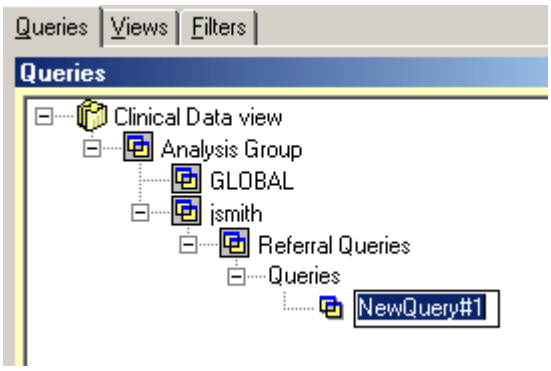
5 DEFINING A QUERY

5.1 Creating a query

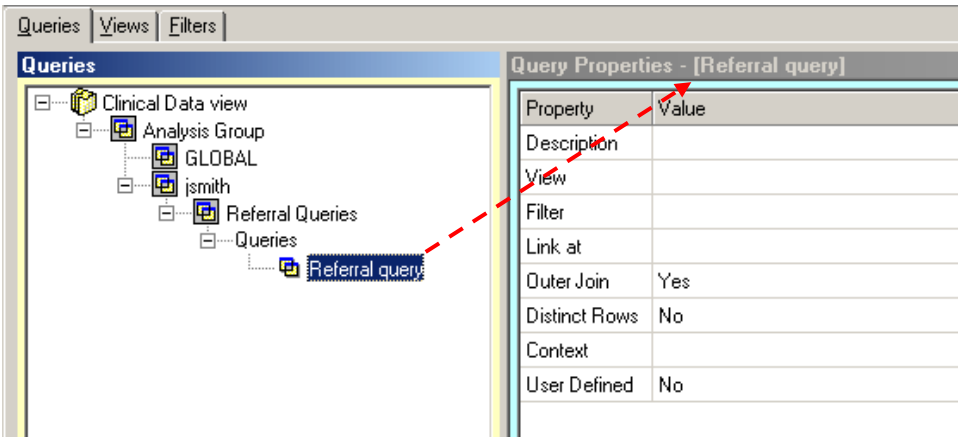
To define a query, go to the **Queries** tab, right click the query group you wish the query to belong to, and choose **New**.



A new query is displayed in the query group. It is represented by the symbol.



Type a name for your query. It is displayed next to the symbol, and the name is also displayed in the titlebar of the main window. You are now ready to define the properties of your query

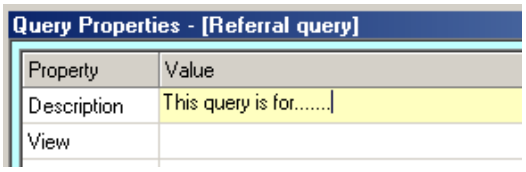


5.2 Adding properties to a query

The following properties can be set in a query.

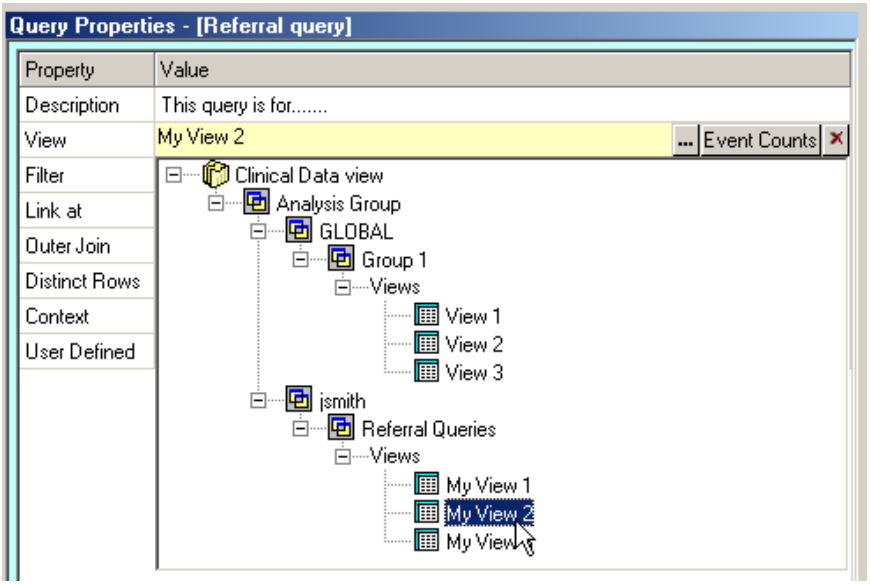
5.2.1 Description

A free text description for reference.

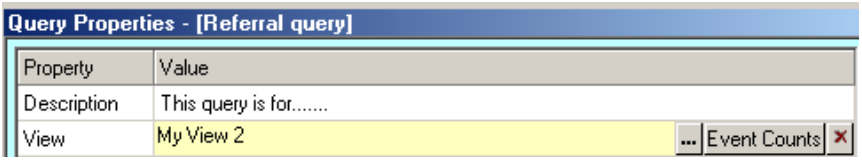


5.2.2 View

To select a view, use the ... button to display a list of the available views. The list displays the Global group and the user's own named group as well as any subgroups, and the views that have been defined within each group. To select a view, double click it.

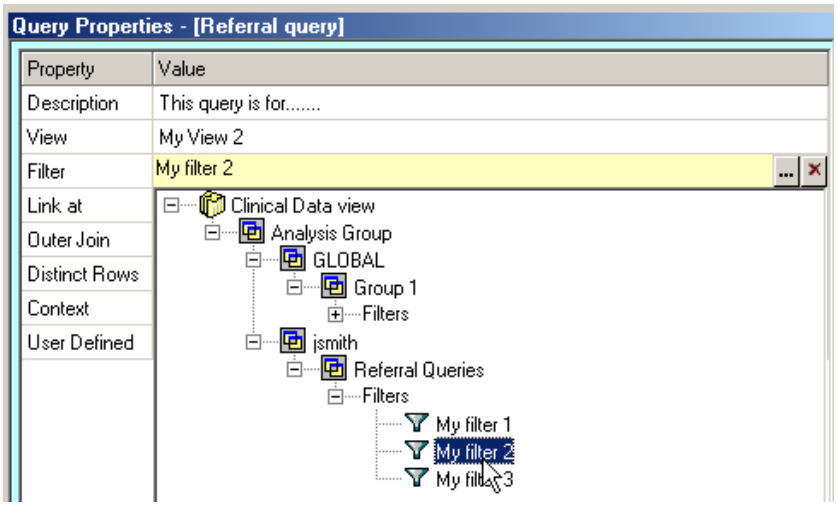


After selecting a view, it is displayed in the **View** field.

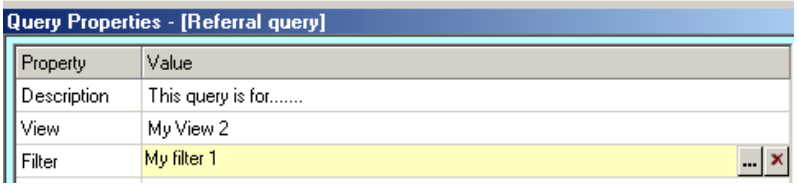


5.2.3 Filter

To select a filter, use the ... button to display a list of the available filters. The list displays the Global group and the user's own named group as well as any subgroups, and the filters that have been defined within each group. To select a filter, double click it.



After double clicking a filter, it is displayed in the **Filter** field.



5.2.4 Link at

The **link level** is used where the filter contains criteria from multiple events. The link level is the level in the design tree beneath which the filter criteria must be met. By default, InfoFlex sets the lowest possible common parent as the link level.

See section **7.2 About Linking** for further details.

5.2.5 Outer Join

The **Join** controls whether or not a subject's events can be retrieved when the view contains items from several different events and not all of the events used in the view exist for that subject. (This is in addition to the criteria defined in the filter.)

When **Outer Join** is set to **Yes**, a subject will be returned if they satisfy the criteria of the filter **and** as long as at least one of the events represented in the view exists for that subject.

When **Outer Join** is set to **No** (ie **Inner Join** is set), a subject is only returned if they satisfy the criteria of the filter **and** if every event used in the view exists for that subject. (Note that it is the **event** that must exist - data does not have to exist in every item used in the view as long as every event used in the view exists).

When a query is first created, if the selected query view or event view only has one event represented, the join type will default to **Inner Join**. Otherwise, **Outer Join** is set. Note that once the join has been set, it will not subsequently change automatically. This is true whether the query view or event view is changed, or whether the query view itself is edited.

See section **7.4 Joining** for further details.

5.2.6 Distinct Rows

In some circumstances, the combination of the link level and the data items being returned will cause the same record to be returned more than once in result set. Setting **Distinct Rows** to **Yes** prevents this happening.

By default, **Distinct Rows** is set to **No**.

See section **7.3 Distinct Rows** for further details.

5.2.7 Context

The Context parameter is relevant when queries are used in documents and reports. The Context parameter sets a lowest common parent for the data that is returned in the document. See the Report Definition and Document definition user guides for further information.

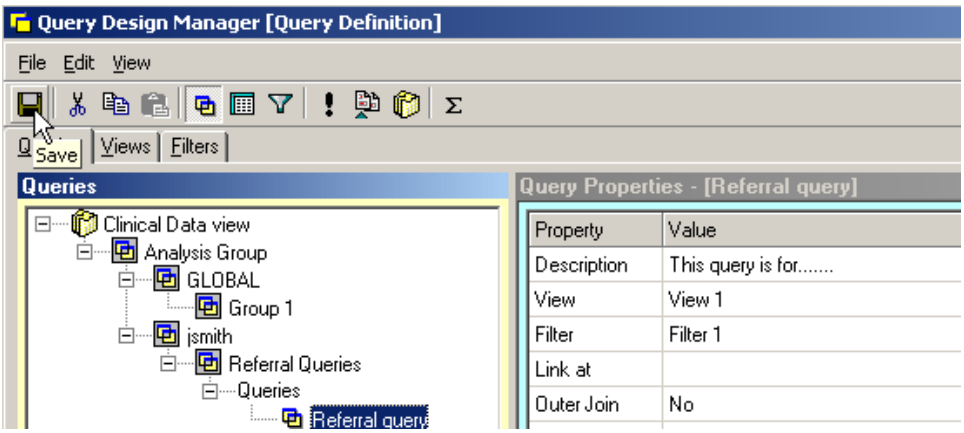
5.2.8 User-defined

User-Defined allows users to write their own SQL queries rather than defining views and filters.

See section **7.6 User-defined** for further details.

5.3 Running a query

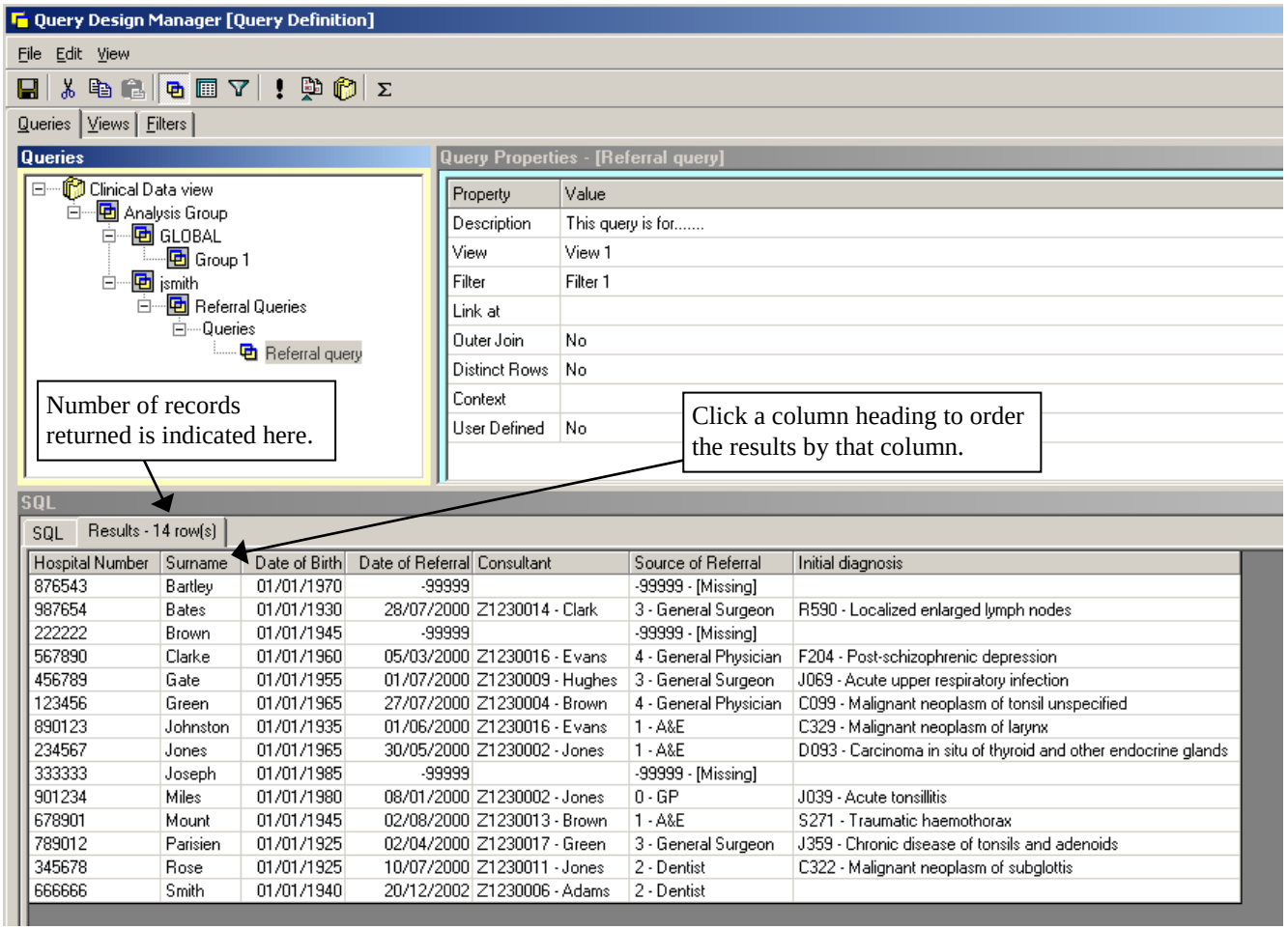
After defining a query, save it by pressing **F5** or the **Save** button.



To view the query results, press the **!** **Run** button on the toolbar.

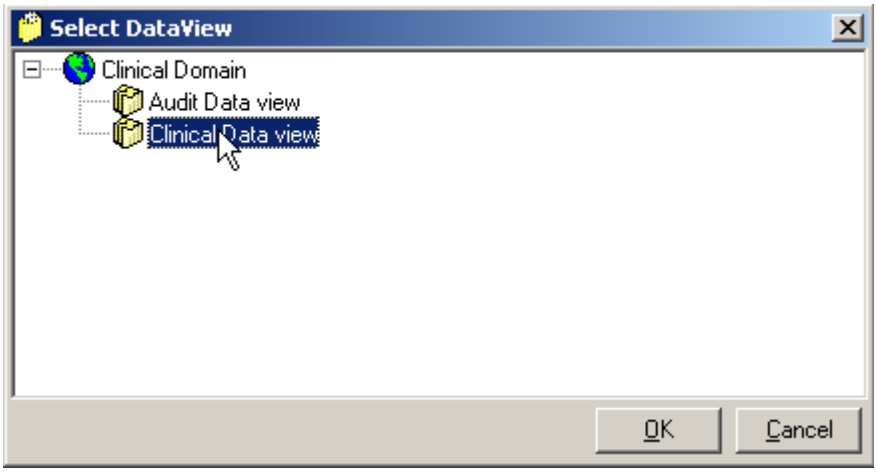
Note that if you make any changes to a query (for instance selecting a different view or filter, or changing the Link or Join), and then run the query without saving, there will be a prompt to save. If you save, the new saved query will be run, and if you do not save, the **unsaved** query will be run rather than the saved query.

The query results are displayed in the **Results** grid below the **Queries** tab.



5.3.1 Studies and queries

When a query is run from a domain, you are prompted for a data view to run it against. The data view filters the results so that only those events that are flagged as belonging to any of the studies in the data view's study list are returned. Therefore if events represented in the view exist for a patient but the events do not belong to any studies represented in the selected data view, those events will not be returned in the query results.

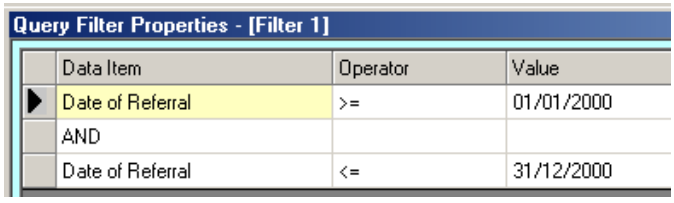


5.3.2 Testing query results

When you are defining a query for use for example in a report, you should always run it in QDM first to ensure that the query is returning the results you expect.

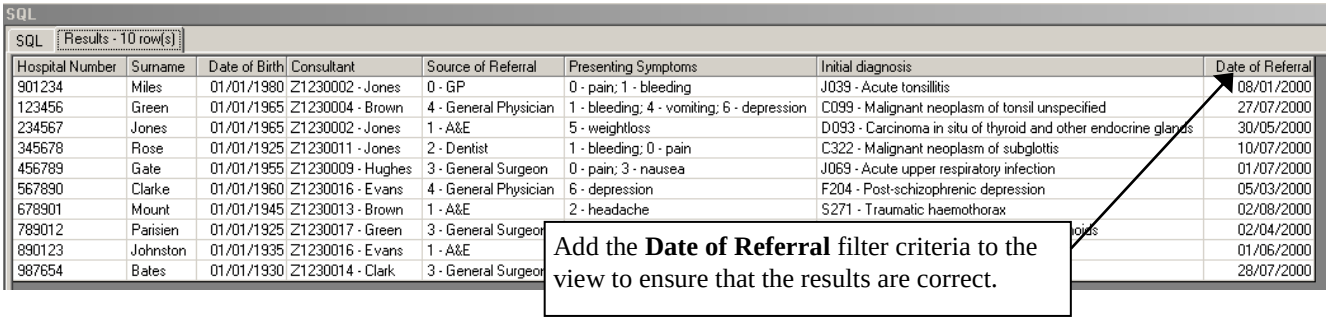
For testing purposes, it can be helpful to include in your view the items that are being used in the filter, even if those items are not needed in the final version of the query.

For example, your query might be returning a list of patients who were referred within a certain time period.



Data Item	Operator	Value
Date of Referral	>=	01/01/2000
AND		
Date of Referral	<=	31/12/2000

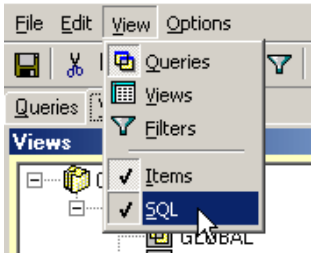
The report may not need to display the **Date of Referral** for each patient, but while you are testing your query it is useful to add the **Date of Referral** to the view so that you can ensure the results are correct. Once the query is producing the correct results, you can remove the **Date of Referral** from the view.



Hospital Number	Surname	Date of Birth	Consultant	Source of Referral	Presenting Symptoms	Initial diagnosis	Date of Referral
901234	Miles	01/01/1980	Z1230002 - Jones	0 - GP	0 - pain; 1 - bleeding	J039 - Acute tonsillitis	08/01/2000
123456	Green	01/01/1965	Z1230004 - Brown	4 - General Physician	1 - bleeding; 4 - vomiting; 6 - depression	C099 - Malignant neoplasm of tonsil unspecified	27/07/2000
234567	Jones	01/01/1965	Z1230002 - Jones	1 - A&E	5 - weightloss	D093 - Carcinoma in situ of thyroid and other endocrine glands	30/05/2000
345678	Rose	01/01/1925	Z1230011 - Jones	2 - Dentist	1 - bleeding; 0 - pain	C322 - Malignant neoplasm of subglottis	10/07/2000
456789	Gate	01/01/1955	Z1230009 - Hughes	3 - General Surgeon	0 - pain; 3 - nausea	J069 - Acute upper respiratory infection	01/07/2000
567890	Clarke	01/01/1960	Z1230016 - Evans	4 - General Physician	6 - depression	F204 - Post-schizophrenic depression	05/03/2000
678901	Mount	01/01/1945	Z1230013 - Brown	1 - A&E	2 - headache	S271 - Traumatic haemothorax	02/08/2000
789012	Parisien	01/01/1925	Z1230017 - Green	3 - General Surgeon			02/04/2000
890123	Johnston	01/01/1935	Z1230016 - Evans	1 - A&E			01/06/2000
987654	Bates	01/01/1930	Z1230014 - Clark	3 - General Surgeon			28/07/2000

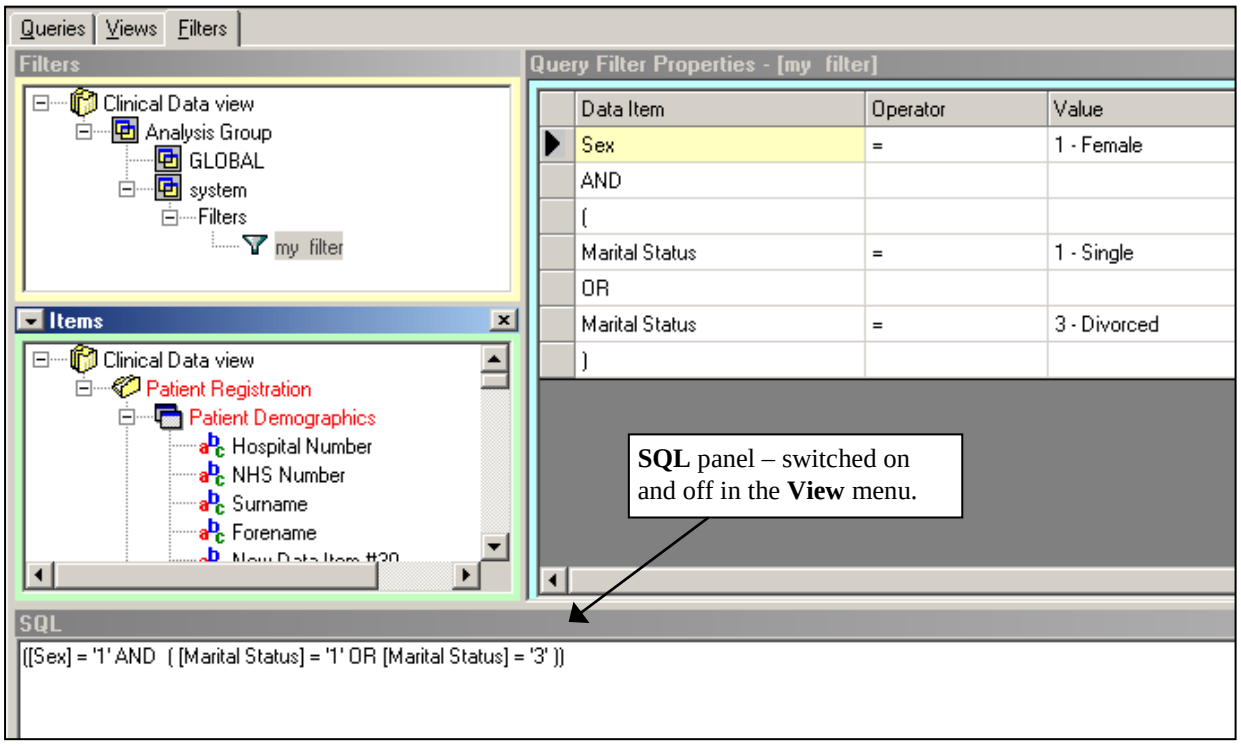
5.3.3 Viewing SQL

You can view the SQL that is created by your view, filter or query definition.

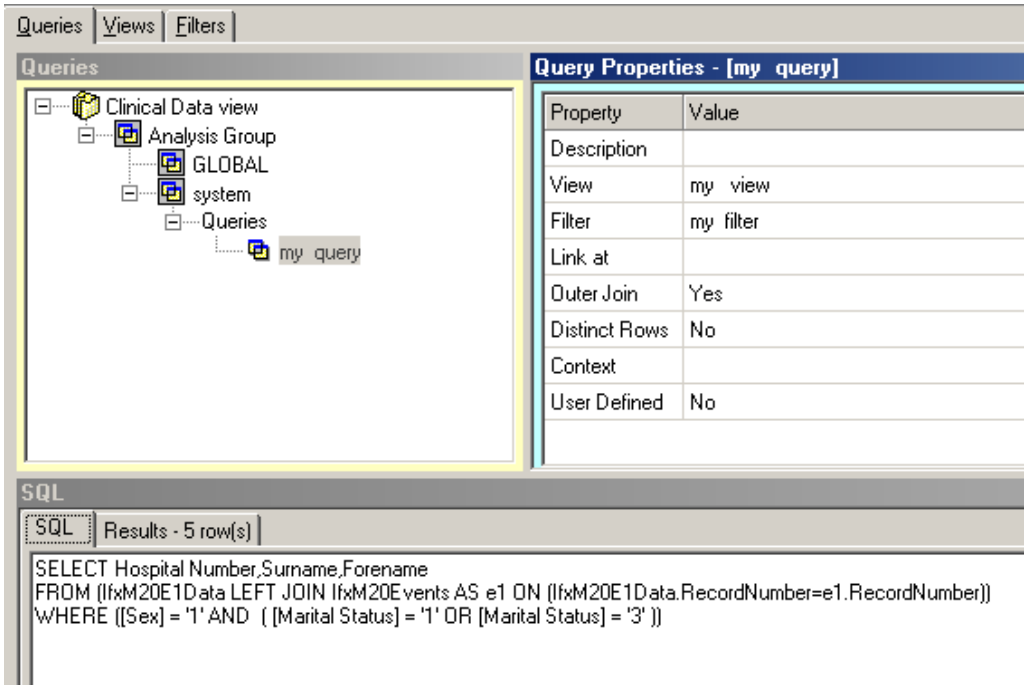


The **View** menu has an **SQL** option which is available whether you are working in the **Queries**, **Views** or **Filters** tab.

When this option is switched on, an **SQL** panel is displayed at the bottom of the tab and the SQL represented by the current view, filter or query is displayed. Note that you should save your view, filter or query to ensure that the SQL is up to date.



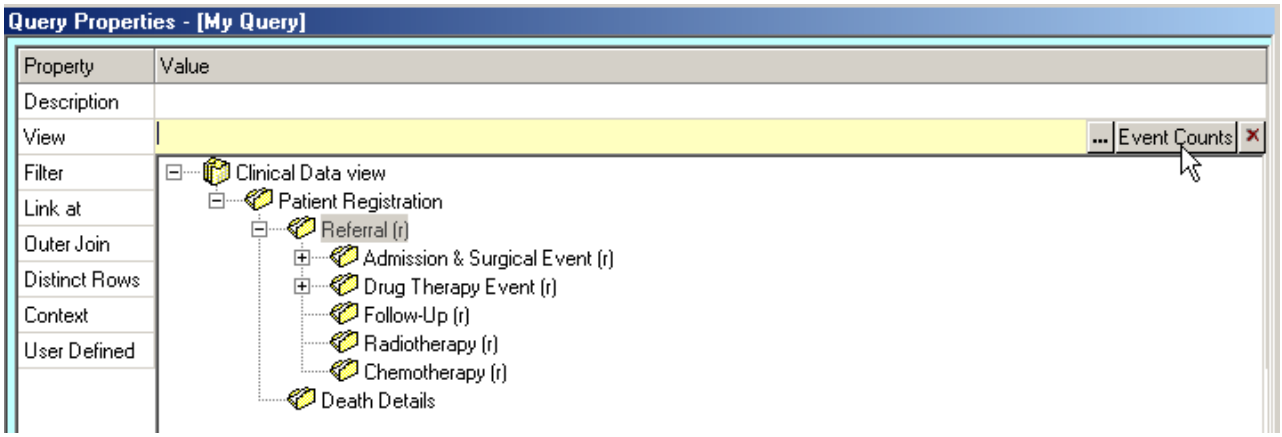
On the **Queries** tab, the SQL is always available when a query has been run. An **SQL** tab is displayed next to the **Results** tab.



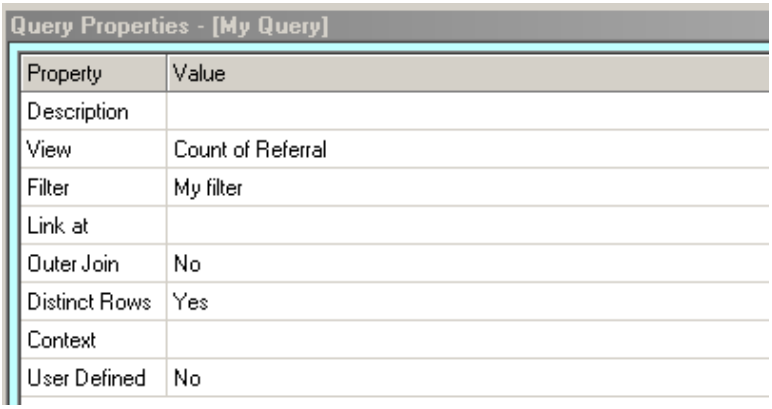
5.4 Event counting

When defining a query, instead of selecting a view, you can choose to count the number of events that meet the filter criteria. This is done by using the **Event Counts** function.
Instead of returning data items, the query will simply count how many occurrences of the selected event match the filter criteria.

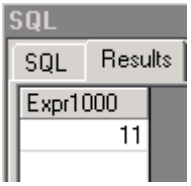
In the **View** selection box, press the **Event Counts** button instead of using the dropdown list of views. The event tree is displayed and you may select any event.



On saving, the View property displays **Count of** followed by the name of the event selected.
Also on saving, **Distinct Rows** is automatically set to **Yes**



When this query is run, one value is returned, which is the number of events that match the filter criteria.



5.5 Exercise

1 In **training group 1,** create a query called **Referral query**.

Set **Referral view** as the view.

Set **Referral filter** as the filter.

Save and run the query.
Note that since Date of Referral is in the view so you can easily check that the right data is being returned (the filter is Date of referral > 1/1/2000).

2 View the SQL on the **Queries**, **Views** and **Filters** tabs.

3 Make a copy of the query in the same group.

In the copy, set the view to **Count** of the **Referral** event.

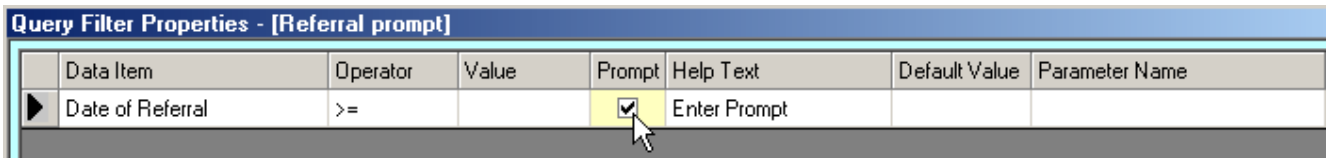
Save and run the query. One row is returned which is the number of Referral events that meet the filter criteria.

6 PROMPT FILTERS

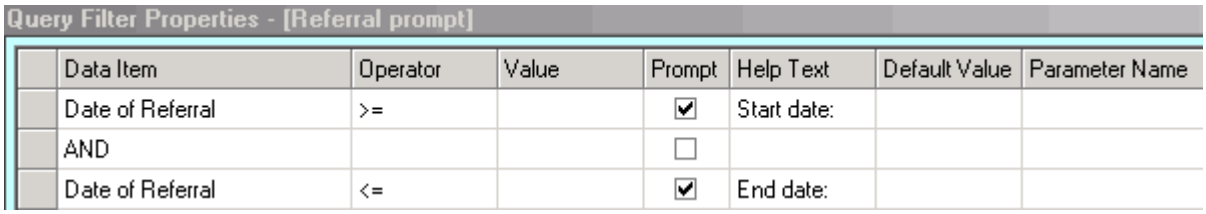
6.1 Defining a prompt filter

Prompts in filters allow the user to specify filter values when the query is run, rather than having to continually update the value specified in the filter itself. For example if a weekly report is run the user can be prompted to enter the date range every time the query is run rather than editing the filter every week, or if the same report is required for several different consultants, one prompt filter can be defined and the user can be prompted for the consultant name on running the query rather than defining one query for each consultant.

To define a prompt filter, create a filter and add an item and operator as usual. Instead of entering a value in the **Value** column, tick the check box in the **Prompt** column.



A filter used in a report which run for a prompted date range might look like this. The user is prompted for the start date and the end date.



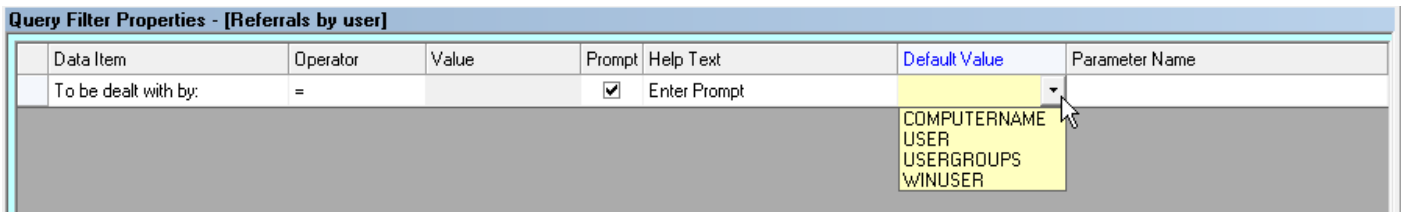
Help Text

The text in the **Help Text** column is displayed when the query is run to indicate to the user the value that is required. You can specify the text that is displayed.

Default value

If you wish the prompted field be populated with a default value, enter it in the **Default Value** column. This value can be over-written by the user when the query is run.

The Information functions USER, COMPUTERNAME, WINGROUPS and WINUSER can be used as default values.



For example, if USER is selected as the default value, then when the query is run, InfoFlex will still prompt the user, but the default value of the current user's username will be supplied as the prompt value, so that the user can just OK the prompt without changing the value. If they need to supply a different value from the default then they are able to change the value as normal.

The prompts will work in Worklist, Data Analysis, Reporting, Documents, Subject Search Queries, etc. They also work with add-ins such as the Summary Doc Addin and the Correspondence Addin and Extract Addin.

The batch process (with the profile that generates/prints/emails reports) does not allow prompting. Although default values are usually used automatically, the information functions above are user-based and will not work here.

The Multiple-Reports add-in is not able to make use of these default values.

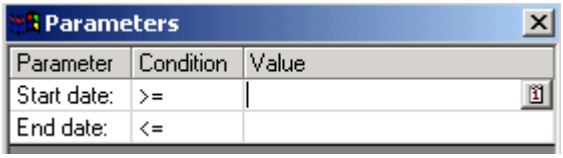
Parameter name

The **Parameter Name** column is for use if the prompted value needs to be printed on a report. The parameter name is included as a document item on the document template, and is substituted for the prompted value when the report is generated. (See section 6.1 of the Report Definition manual).

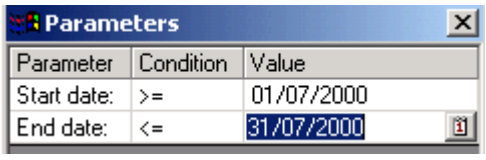
The **Parameter Name** column can also be used if the same parameter is used more than once in the filter. If the same parameter name is used against each instance of the parameter, it will only be prompted for once. The entered value will automatically be used in each subsequent occurrence of the parameter without further prompting.

Running a query containing a prompt filter

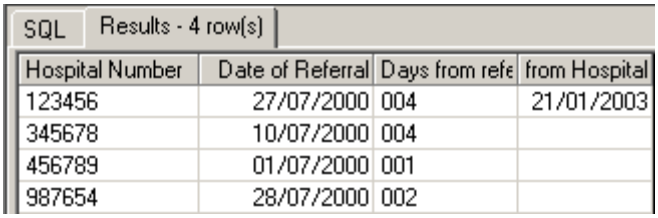
When a query containing a prompt filter is run, a **Parameters** box is displayed listing all the items that have been set as prompt items. The **Parameter** column contains the help text that has been entered for the item.



Enter values in the **value** column for both of the items then press OK.



The results are returned in the **Results** grid as usual.



6.2 Exercise

1 Create a filter in **Training group 1** called **Prompt filter**.
Set it to prompt for a date range for Date of Referral (ie prompt for earliest and latest Date of Referral).
Set prompt text for each parameter.
Save the filter.

2 Create a new query in the **Training group 1** called **Prompt query**.
Select **Referral View** and **Prompt filter**.
Save and run the query.
Enter dates of 1/1/2000 and today.
Then rerun the query with dates of 1/7/2000 and 31/7/2000.

7 QUERY PARAMETERS

7.1 Types of query

Simple queries use data from one off events, or data from the first level of a repeat event.

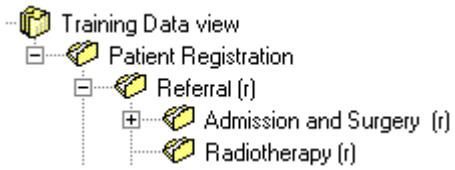
Complex Queries can use Complex Views and Complex Filters.
Complex Views can include data items from several events (including repeat events) at several levels.
Complex Filters can include criteria from different events.
In the case of Complex Queries, different results can be returned depending on the way the **Linking and Joining parameters** are set in the query**.**

7.2 About Linking

Where filter criteria are taken from multiple events, the **Link Level** controls the way the filter criteria are applied to the **View** and hence affects which set of data is returned.

The **Link Level** is the level in the design tree beneath which the filter criteria must be met. The event at which the link level is set is known as the **common parent**. The filter criteria must be met in child events of that common parent.

For example. In this design, filter criteria might be defined using items from the **Admission & Surgical** event and from the **Radiotherapy** event.

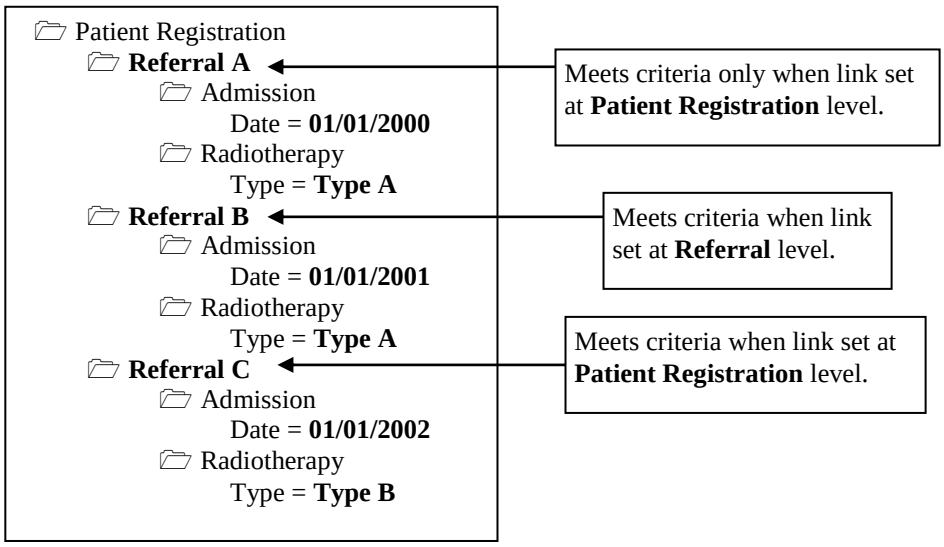


The filter might be to return all Referrals where Date of Admission >= 01/01/2001 (from the **Admission** event) and Radiotherapy = type A (from the **Radiotherapy** event).

Query Filter Properties - [Link example]

	Data Item	Operator	Value
	Date of Admission	>=	01/01/2001
	AND		
	Type of radiotherapy	=	0 - Type A

Consider this subject overview:



The link level specifies whether these criteria must be fulfilled within the same **Referral** event, or whether the criteria can be fulfilled across different occurrences of the **Referral** event.

When determining which Referral events meet the criteria:

- if the link level is set at the **Referral** event, the Referral event is the **Common Parent** and in order for a particular referral event to be returned, the filter criteria must all be met in child events of that referral. In the example above, the criteria are only met within Referral B.

- if the link level is set at the **Patient Demographics** event, the Patient Demographics event is the Common Parent and Referral events can be returned as long as the filter criteria are met somewhere within the patient. In the example above, all the Referral events would be returned.

So, with the link level set at **Referral**, only one record is returned.

SQL | Results - 1 row(s)

Hospital Number	Appointment date	Date of Admission	Type of radiotherapy
LINK TEST PATIENT	01/01/2001	01/01/2001	0 - Type A

But with the link level set at **Patient Registration**, 12 records are returned. This is because there are four combinations of ways that the filter criteria can be met within the subject overview, and these four ways are listed once for each of the three Referral events.

SQL | Results - 12 row(s)

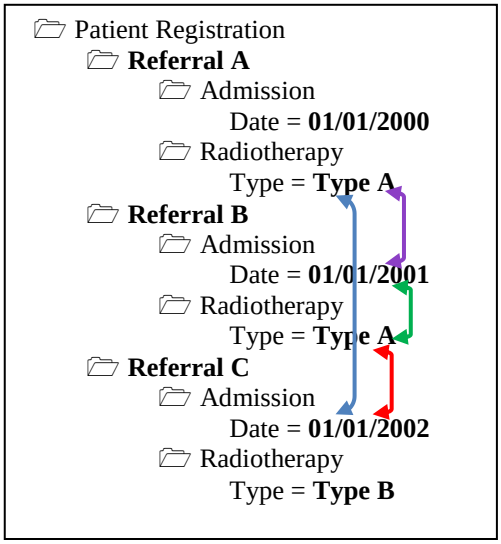
Hospital Number	Appointment date	Date of Admission	Date of this radiotherapy	Type of radiotherapy	
LINK TEST PATIENT	01/01/2000	01/01/2001	01/01/2000	0 - Type A	●
LINK TEST PATIENT	01/01/2000	01/01/2002	01/01/2000	0 - Type A	●
LINK TEST PATIENT	01/01/2000	01/01/2001	01/01/2001	0 - Type A	●
LINK TEST PATIENT	01/01/2000	01/01/2002	01/01/2001	0 - Type A	●
LINK TEST PATIENT	01/01/2001	01/01/2001	01/01/2000	0 - Type A	●
LINK TEST PATIENT	01/01/2001	01/01/2002	01/01/2000	0 - Type A	●
LINK TEST PATIENT	01/01/2001	01/01/2001	01/01/2001	0 - Type A	●
LINK TEST PATIENT	01/01/2001	01/01/2002	01/01/2001	0 - Type A	●
LINK TEST PATIENT	01/01/2002	01/01/2001	01/01/2000	0 - Type A	●
LINK TEST PATIENT	01/01/2002	01/01/2002	01/01/2000	0 - Type A	●
LINK TEST PATIENT	01/01/2002	01/01/2001	01/01/2001	0 - Type A	●
LINK TEST PATIENT	01/01/2002	01/01/2002	01/01/2001	0 - Type A	●

Referral A

Referral **B**

Referral C

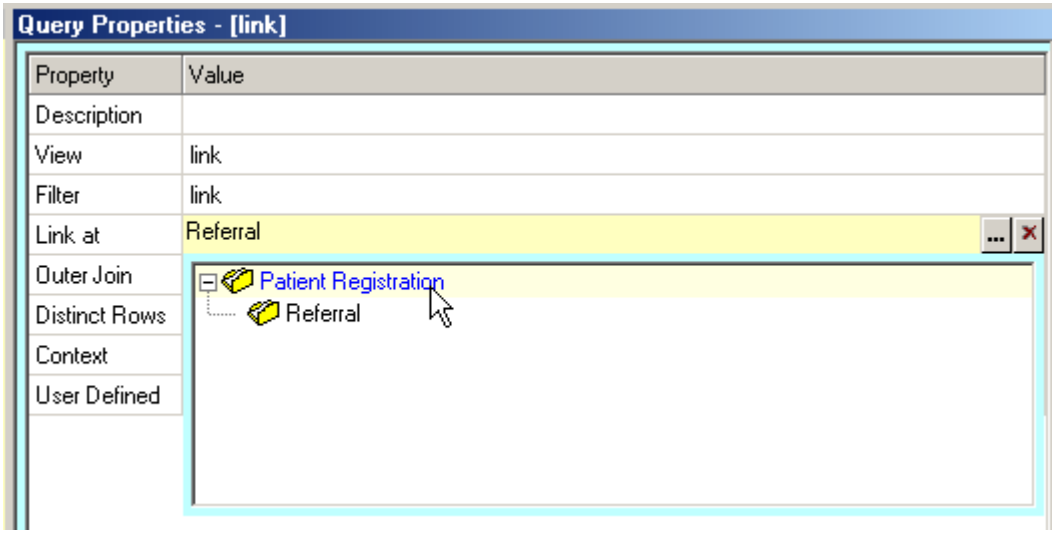
The coloured dots correspond to the ways that the filter criteria are met (illustrated in the subject overview below).
Each way is listed once against each referral event.



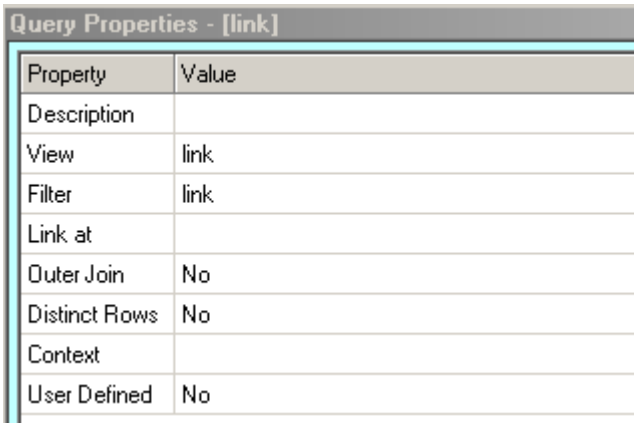
The coloured arrows mark the four ways in which the filter criteria are met across the subject.

7.2.1 How to set the Link Level

To set the Link Level, edit the query and press the ... button in the **Link at** property.
Double click the event which you wish to set as the Common Parent.



By default, no link level is shown within a query definition and InfoFlex uses the lowest possible common parent as the link level.



7.2.2 Exercise

This exercise recreates the example described in section 7.2 above.

In the **Training Data view**, create a view called **Link example** as follows:
From the **Patient Registration** event: Hospital number
From the **Referral** event, **Details** panel: Appointment date
From the **Admission & Surgical** event, **Surgery Details** panel: Date of Admission
From the **Radiotherapy** event: Date of this radiotherapy, Type of Radiotherapy

Create a filter called **Link example** as follows:
Date of Admission >= 01/01/2001 (**Admission & Surgical** event)
AND
Type of radiotherapy = Type A (**Radiotherapy** event)

Create a query called **Link example** and select the **Link example** view and the **Link example** filter.

Set the Link level to the **Referral** event, then save and run the query.

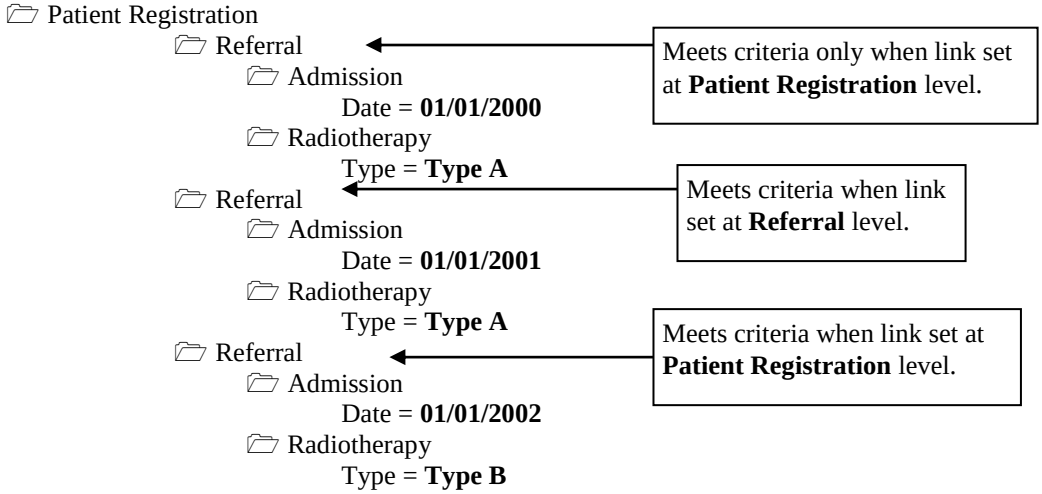
With the link level set at the **Referral** level, the filter criteria must be met within the same Referral event.

1 record is returned, as there is only 1 record where the criteria are met within the same Referral.

Set the Link level to the **Patient Registration** event, then save and run the query.

With the link level set at the Patient Registration level, the filter criteria can be met anywhere within the patient. 12 records are returned. This is because using the Admission and Radiotherapy events from anywhere in the subject overview, there are 4 different ways in which the criteria can be met and these are listed for each of the 3 **Referral** events. Order the grid by **Appointment date** to see this more clearly.

The subject overview for the patient in question is as shown below.
If you wish to review the data, go to Data Entry and in the Training data view, search for the patient with a Hospital Number of **LINK TEST PATIENT**.



7.3 Distinct Rows

As we have seen above, the number of records returned matches the number of combinations in which the filter criteria can be met within the events, regardless of which items are included in the view.

In the example above, with the link level set at Patient Registration, 12 records were returned.
Since the items in the view come from several repeating events at the same level, each record returned represents each different combination of the data.

With the filter described above in the Link example, these results are displayed:

SQL | Results - 12 row(s)

Hospital Number	Appointment date	Date of Admission	Date of this radiotherapy	Type of radiotherapy
LINK TEST PATIENT	01/01/2000	01/01/2001	01/01/2000	0 - Type A
LINK TEST PATIENT	01/01/2000	01/01/2002	01/01/2000	0 - Type A
LINK TEST PATIENT	01/01/2000	01/01/2001	01/01/2001	0 - Type A
LINK TEST PATIENT	01/01/2000	01/01/2002	01/01/2001	0 - Type A
LINK TEST PATIENT	01/01/2001	01/01/2001	01/01/2000	0 - Type A
LINK TEST PATIENT	01/01/2001	01/01/2002	01/01/2000	0 - Type A
LINK TEST PATIENT	01/01/2001	01/01/2001	01/01/2001	0 - Type A
LINK TEST PATIENT	01/01/2001	01/01/2002	01/01/2001	0 - Type A
LINK TEST PATIENT	01/01/2002	01/01/2001	01/01/2000	0 - Type A
LINK TEST PATIENT	01/01/2002	01/01/2002	01/01/2000	0 - Type A
LINK TEST PATIENT	01/01/2002	01/01/2001	01/01/2001	0 - Type A
LINK TEST PATIENT	01/01/2002	01/01/2002	01/01/2001	0 - Type A

If the view only contained items from the Patient Registration and Referral events, 12 records would be returned, and some of the records would look the same since the different data items that distinguish them would not be included in the view. In this case, you can choose to display **Distinct Rows** in order to remove the apparent duplicates.

With the same filter but fewer items in the view, the same records are displayed, but now there are no items from the Admission or Radiotherapy events to distinguish one record from another.

SQL

SQL | Results - 12 row(s)

Hospital Number	Appointment date
LINK TEST PATIENT	01/01/2000
LINK TEST PATIENT	01/01/2000
LINK TEST PATIENT	01/01/2000
LINK TEST PATIENT	01/01/2000
LINK TEST PATIENT	01/01/2001
LINK TEST PATIENT	01/01/2001
LINK TEST PATIENT	01/01/2001
LINK TEST PATIENT	01/01/2001
LINK TEST PATIENT	01/01/2002
LINK TEST PATIENT	01/01/2002
LINK TEST PATIENT	01/01/2002
LINK TEST PATIENT	01/01/2002

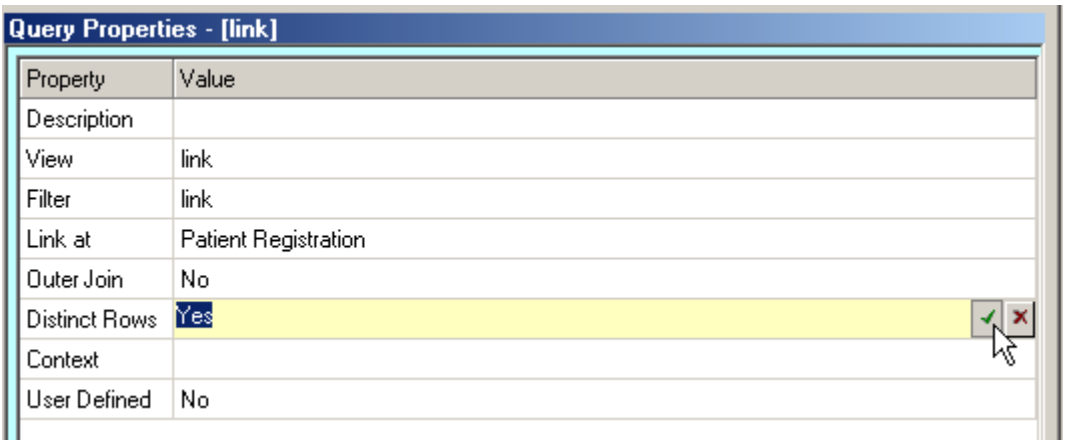
Setting **Distinct Rows** to **Yes** will reduce the results to show distinct records only.

SQL | Results - 3 row(s)

Hospital Number	Appointment date
LINK TEST PATIENT	01/01/2000
LINK TEST PATIENT	01/01/2001
LINK TEST PATIENT	01/01/2002

7.3.1 How to set Distinct Rows

To switch on **Distinct Rows**, edit the query and set the **Distinct Rows** property to **Yes**.



Note that by default, **Distinct Rows** is set to **No**.

7.3.2 Exercise

This exercise recreates the example described in section 7.3above.

Edit the **Link example** view created in the Linking exercise in 7.2.2above.

Delete the last 3 items in the view so that the view only contains **Hospital Number** and **Appointment Date**. Save the view.

Run the **Link Example** query with the link level set at the **Patient Registration** event.

12 records are returned, but since the items from the child events are not present, there appear to be duplicate records.

Set **Distinct Rows** to **Yes**.

Save then rerun the query. The duplicate rows are removed and now only 3 records are returned.

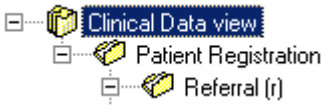
7.4 Joining

Joining affects which set of data is returned when multiple events are represented in a view. This is in addition to the criteria defined in the filter. Joining controls whether or not a record can be retrieved where not all of the events used in the view exist for that subject.

When **Outer Join** is set to **Yes**, a subject will be returned as long as they satisfy the filter criteria **and** as long as at least one event used in the view exists for that subject.

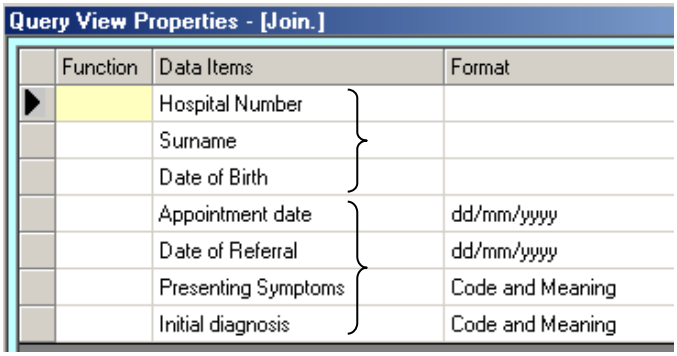
When **Outer Join** is set to **No** (ie **Inner Join** is set), a subject is only returned if every event represented in the view exists for that subject. (Note that it is the **event** that must exist - data does not have to exist in every item used in the view as long as every event used in the view exists. If there is no data in any of the items in the view, a blank row is returned.)

For example: In this design:



a view is created which contains items from both the Patient Registration event and the Referral event.

(In this example, no filter criteria are defined so all patients are available)

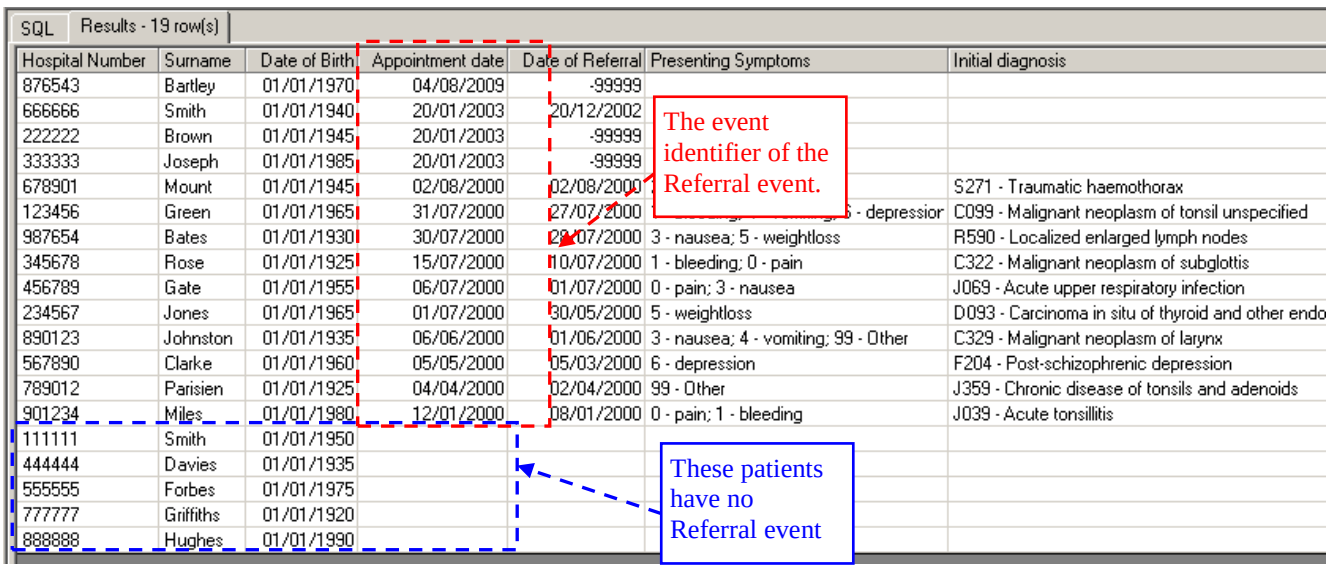


Function	Data Items	Format
	Hospital Number	
	Surname	
	Date of Birth	
	Appointment date	dd/mm/yyyy
	Date of Referral	dd/mm/yyyy
	Presenting Symptoms	Code and Meaning
	Initial diagnosis	Code and Meaning

Outer Join

When this view is run in a query with an **Outer Join (**and no filter criteria), 19 records are returned, however the last 5 records do not show any data against the items from the Referral event. The Appointment Date item is the identifier of the event and it is included in the view so we can be sure that the Referral event does not exist for these 5 records.

The 5 records are included because the **Patient Registration** event which is represented in the view does exist for them, and an **Outer Join** will return a record as long as at least one event represented in the view exists.



Hospital Number	Surname	Date of Birth	Appointment date	Date of Referral	Presenting Symptoms	Initial diagnosis
876543	Bartley	01/01/1970	04/08/2009	-99999		
666666	Smith	01/01/1940	20/01/2003	20/12/2002		
222222	Brown	01/01/1945	20/01/2003	-99999		
333333	Joseph	01/01/1985	20/01/2003	-99999		
678901	Mount	01/01/1945	02/08/2000	02/08/2000		S271 - Traumatic haemothorax
123456	Green	01/01/1965	31/07/2000	27/07/2000	- depressior	C099 - Malignant neoplasm of tonsil unspecified
987654	Bates	01/01/1930	30/07/2000	28/07/2000	3 - nausea; 5 - weightloss	R590 - Localized enlarged lymph nodes
345678	Rose	01/01/1925	15/07/2000	10/07/2000	1 - bleeding; 0 - pain	C322 - Malignant neoplasm of subglottis
456789	Gate	01/01/1955	06/07/2000	01/07/2000	0 - pain; 3 - nausea	J069 - Acute upper respiratory infection
234567	Jones	01/01/1965	01/07/2000	30/05/2000	5 - weightloss	D093 - Carcinoma in situ of thyroid and other endo
890123	Johnston	01/01/1935	06/06/2000	01/06/2000	3 - nausea; 4 - vomiting; 99 - Other	C329 - Malignant neoplasm of larynx
567890	Clarke	01/01/1960	05/05/2000	05/03/2000	6 - depression	F204 - Post-schizophrenic depression
789012	Parisien	01/01/1925	04/04/2000	02/04/2000	99 - Other	J359 - Chronic disease of tonsils and adenoids
901234	Miles	01/01/1980	12/01/2000	08/01/2000	0 - pain; 1 - bleeding	J039 - Acute tonsillitis
111111	Smith	01/01/1950				
444444	Davies	01/01/1935				
555555	Forbes	01/01/1975				
777777	Griffiths	01/01/1920				
888888	Hughes	01/01/1990				

Inner Join

However when the same view is run in the same query (and no filter criteria) but with an **Inner Join**, only 14 records are returned. The 5 records with no Referral event are omitted because Inner Join specifies that a record can only be returned when all events represented in the view exist.

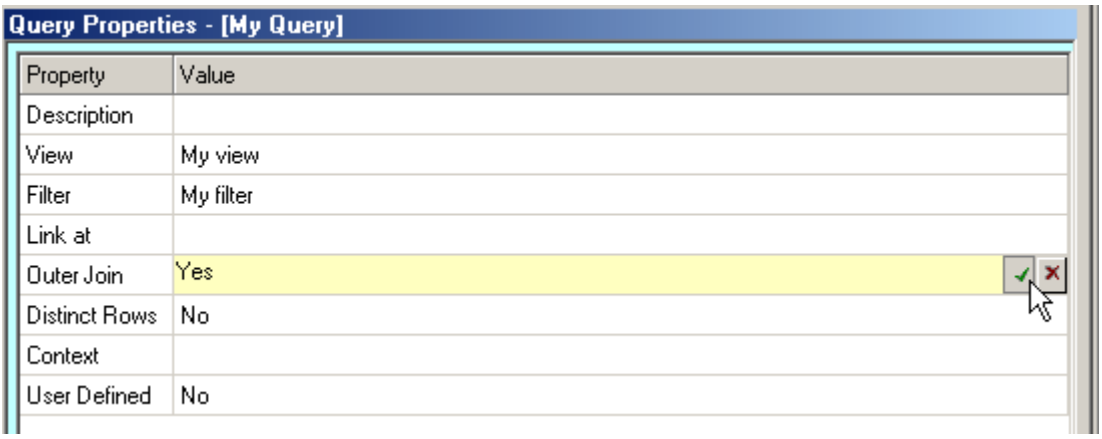
SQL | Results - 14 row(s)

Hospital Number	Surname	Date of Birth	Appointment date	Date of Referral	Presenting Symptoms	Initial diagnosis
901234	Miles	01/01/1980	12/01/2000	08/01/2000	0 - pain; 1 - bleeding	J039 - Acute tonsillitis
789012	Parisien	01/01/1925	04/04/2000	02/04/2000	99 - Other	J359 - Chronic disease of tonsils and adenoids
567890	Clarke	01/01/1960	05/05/2000	05/03/2000	6 - depression	F204 - Post-schizophrenic depression
890123	Johnston	01/01/1935	06/06/2000	01/06/2000	3 - nausea; 4 - vomiting; 99 - Other	C329 - Malignant neoplasm of larynx
234567	Jones	01/01/1965	01/07/2000	30/05/2000	5 - weightloss	D093 - Carcinoma in situ of thyroid and other end
456789	Gate	01/01/1955	06/07/2000	01/07/2000	0 - pain; 3 - nausea	J069 - Acute upper respiratory infection
345678	Rose	01/01/1925	15/07/2000	10/07/2000	1 - bleeding; 0 - pain	C322 - Malignant neoplasm of subglottis
987654	Bates	01/01/1930	30/07/2000	28/07/2000	3 - nausea; 5 - weightloss	R590 - Localized enlarged lymph nodes
123456	Green	01/01/1965	31/07/2000	27/07/2000	1 - bleeding; 4 - vomiting; 6 - depression	C099 - Malignant neoplasm of tonsil unspecified
678901	Mount	01/01/1945	02/08/2000	02/08/2000	2 - headache	S271 - Traumatic haemothorax
666666	Smith	01/01/1940	20/01/2003	20/12/2002		
222222	Brown	01/01/1945	20/01/2003	-99999		
333333	Joseph	01/01/1985	20/01/2003	-99999		
876543	Bartley	01/01/1970	04/08/2009	-99999		

7.4.1 How to set the Join

To set the query to Outer Join, edit the query and set the **Outer Join** property to **Yes**.

To set the query to Inner Join, edit the query and set the **Outer Join** property to **No**.



Default Outer Join setting

When a query is first created, if the selected query view or event view only has only one event represented, the join type will default to **Inner Join**. Otherwise, **Outer Join** is set. Note that once the join has been set, it will not subsequently change automatically. This is true whether the query view or event view is changed, or whether the query view itself is edited.

7.4.2 Exercise

This exercise recreates the example described in section 7.4above.

In the **Clinical Data view**, create a new filter in the **Training group 1** called **All patients**.
Set the criteria to **Hospital Number IS NOT EMPTY**.

In the **Clinical Data view**, create a new query in the **Training group 1** called **Join Example**.
Select the **Referral View** view and the **All patients** filter.

Save the query with **Outer Join** set to **Yes**.

Run the query.
20 records are returned.
Order the records by **Appointment Date** and note that some of the records have no Appointment Date. Since Appointment Date is the event identifier of the Referral event, this means that those records have no referral event.

Set **Outer Join** to **No**. **Save** then re-run the query.

This time 15 records are returned. This is because records are only returned if all the events that are represented in the view exist.

7.5 Context

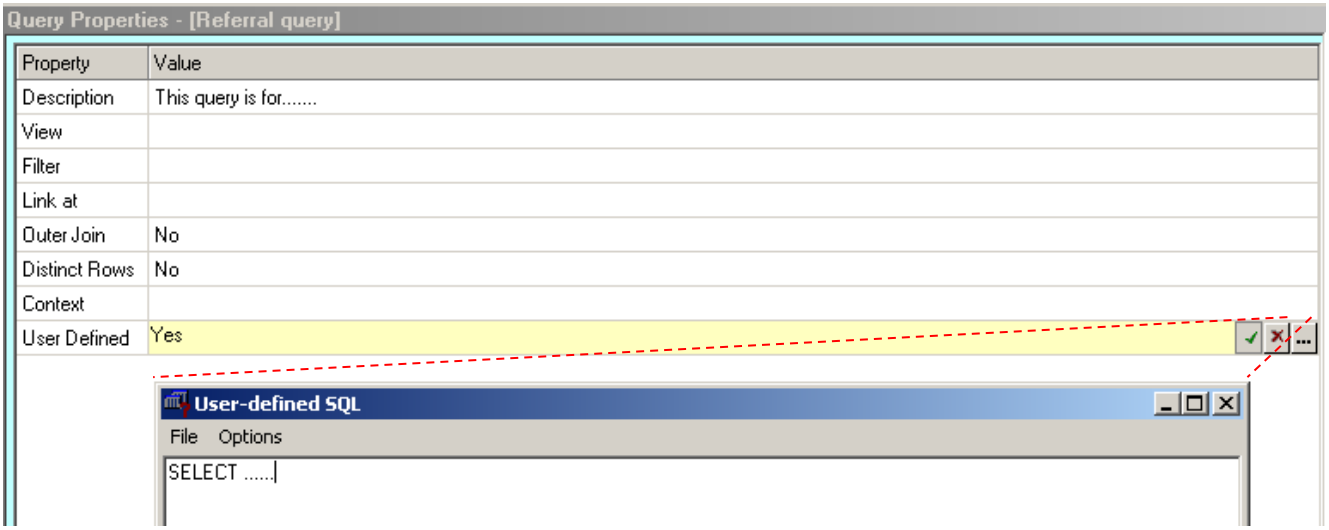
The Context parameter is relevant when queries are used in documents and reports. The Context parameter sets a lowest common parent for the data that is returned in the document. See the Report Definition and Document definition user guides for further information.

7.6 User-defined

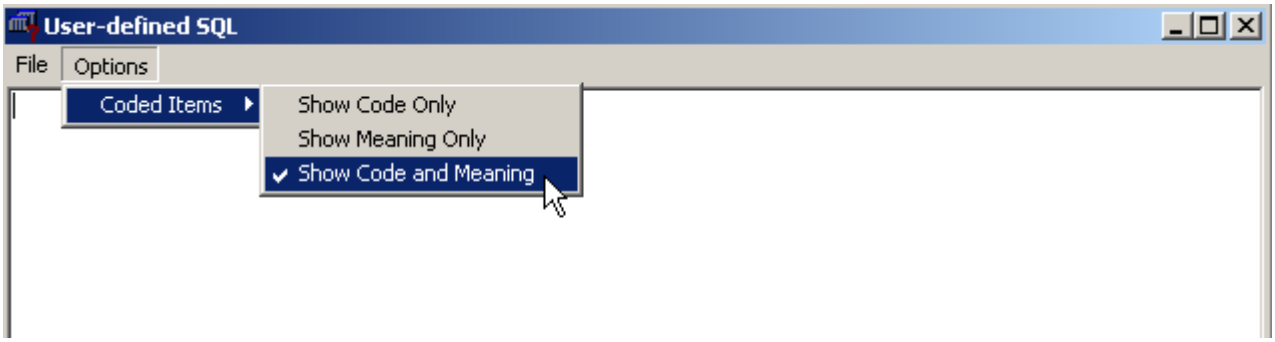
Querying functions that are not currently available in QDM can be requested through the CIMS helpdesk. Agreed functions that are developed are released to all customers. However, in the event of an urgent requirement, users may write their own SQL instead of using InfoFlex views and filters. Please contact CIMS helpdesk if you think you need to use this function.

If it is agreed that user-defined SQL is required, set **User-Defined** to **Yes** and press the ... button to display. A **User-defined SQL** box is displayed in which to enter your own SQL.

Note that on setting **User-Defined** to **Yes**, any selected view and filter are removed.



Note that where a coded item is included in a SELECT statement, the **Coded Items** option on the **Options** menu controls the format that the data is returned in.

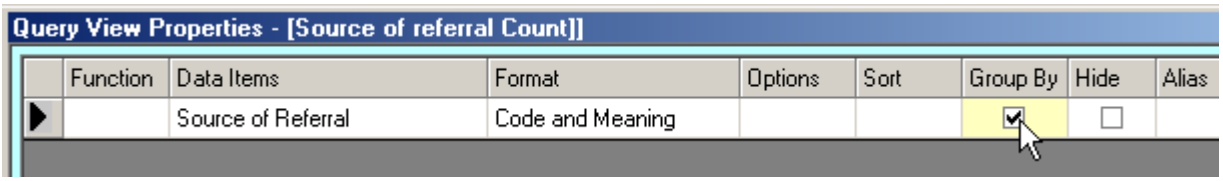


8 VIEW PARAMETERS – OCCURRENCE COUNTING

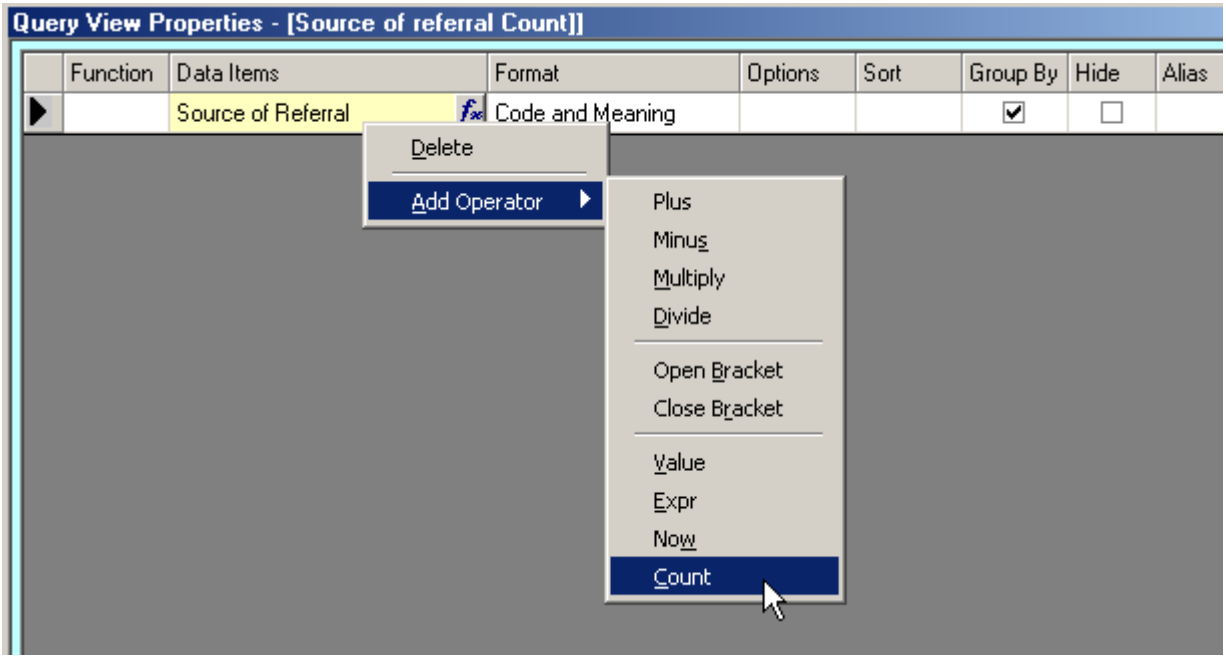
8.1 Occurrence Counting

Occurrence counting counts the number of times that each value entered in an item occurs. This function can be used, for example to count how many times each source of referral has occurred. Occurrence counting has to be used with the **Group by** function.

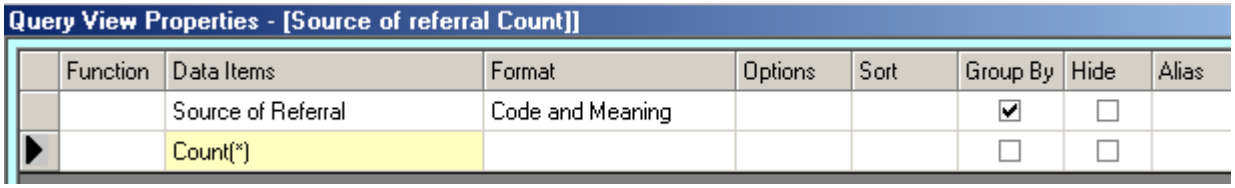
To count the number of times that each item value occurs, add the item that you wish to count to a view and tick the **Group By** column.



Right click the item and choose **Add Operator** then **Count**.



A second row displays the **Count** command.



When this view is run in a query, the results show the number of times each value entered in the Source of Referral item has been used.

SQL | Results - 6 row(s)

Source of Referral	Count
-88888 - [Unknown]	1
-99999 - [Missing]	2
0 - GP	6
1 - A&E	3
2 - Dentist	1
3 - General Surgeon	3

This function can be used with coded, boolean, dictionary, date, text and value items.

SQL | Results - 6 row(s)

Hospital Name	Count
-88888 - [Unknown]	2
-99999 - [Missing]	1
AB100 - St Matthew's Hospital	2
BC200 - St Mark's Hospital	4
CD300 - St Luke's Hospital	2
DE400 - St John's Hospital	3

This function can also be used with more than one item to show the number of times a combination of values occurs.
Each item must have the **Group By** option ticked.

Query View Properties - [Source of referral Count]]

Function	Data Items	Format	Options	Sort	Group By	Hide	Alias
	Hospital Name	Code and Meaning			☑	☐	
	Source of Referral	Code and Meaning			☑	☐	
	Count(*)				☐	☐	

The results show the number of times each combination of values has been entered.

SQL | Results - 10 row(s)

Hospital Name	Source of Referral	Count
-88888 - [Unknown]	1 - A&E	1
AB100 - St Matthew's Hospital	0 - GP	2
AB100 - St Matthew's Hospital	3 - General Surgeon	1
BC200 - St Mark's Hospital	0 - GP	3
BC200 - St Mark's Hospital	1 - A&E	2
BC200 - St Mark's Hospital	2 - Dentist	1
CD300 - St Luke's Hospital	0 - GP	2
DE400 - St John's Hospital	-99999 - [Missing]	1
DE400 - St John's Hospital	1 - A&E	2
DE400 - St John's Hospital	3 - General Surgeon	1

8.2 Exercise

In the **Clinical Data View**, in **Training group 1**, create a view called **Occurrence counting**.

Add the **Source of Referral** (**Referral** event, **Details** panel) and set the format to **Code and Meaning** and tick the **Group By** column.

In the second row of the view, add the **Count** operator from the right click menu.

Save the view.

In **Training group 1**, create a query called **Occurrence Counting**.

Select the **Occurrence Counting** view and the **All patients** filter.

Run the query.

These results should be returned:

SQL | Results - 6 row(s)

Source of Referral	Count
-99999 - [Missing]	4
0 - GP	1
1 - A&E	3
2 - Dentist	2
3 - General Surgeon	3
4 - General Physician	2

Add the **Category** item (**Referral** event, **Details** panel) to the **Occurrence Counting** view.
Set the format to **Code and Meaning** and tick the **Group By** column.
Move **Category** to be the second row in the view.
Save the view.

Rerun the **Occurrence Counting** query.

These results should be returned:

SQL | Results - 10 row(s)

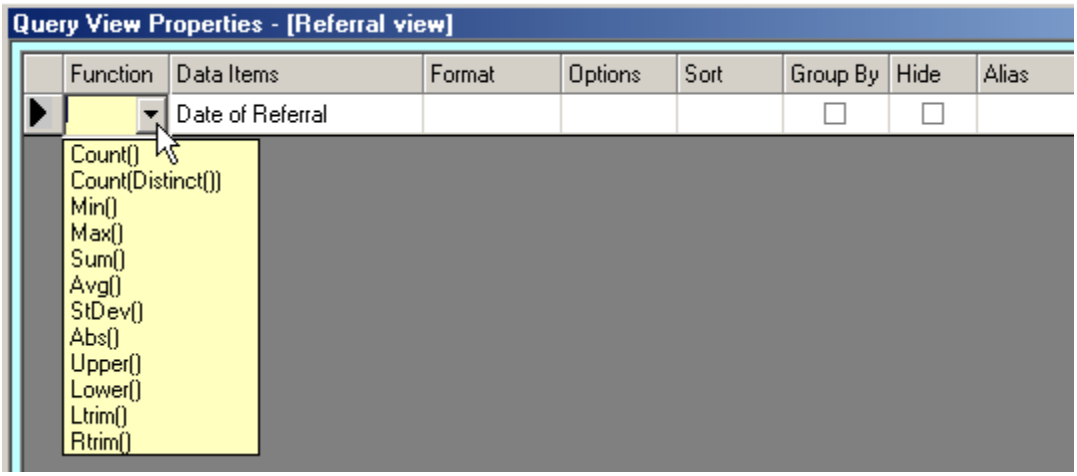
Source of Referral	Category	Count
-99999 - [Missing]		3
-99999 - [Missing]	1 - Private	1
0 - GP	0 - NHS	1
1 - A&E	1 - Private	2
1 - A&E	2 - Overseas visitor	1
2 - Dentist	0 - NHS	1
2 - Dentist	2 - Overseas visitor	1
3 - General Surgeon	0 - NHS	1
3 - General Surgeon	2 - Overseas visitor	2
4 - General Physician	0 - NHS	2

9 MANIPULATING DATA IN VIEWS

9.1 Functions

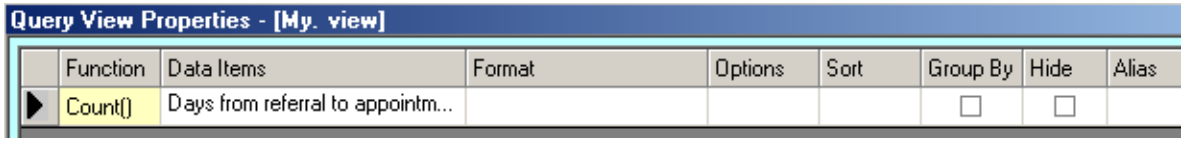
The functions described in this section perform aggregate or manipulative tasks on the data items in the **View**.

In all cases, to add these functions to the view, first add the item to the view, then select the function from the dropdown list in the **Function** column.

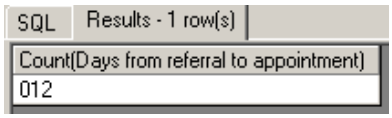


9.1.1 Count()

The **Count()** function counts the number of records which have a value recorded in the field chosen. This example produces a count of how many patients matching the **filter** criteria have something recorded in the field **Days from Referral to Appointment**



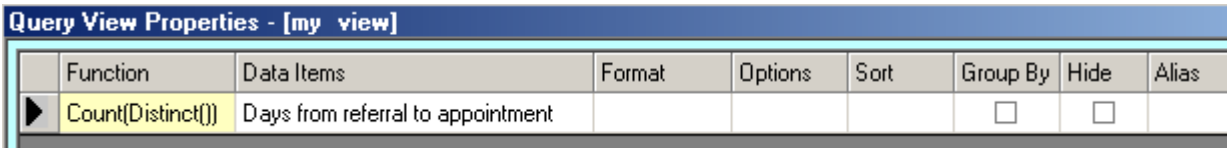
One row is returned:



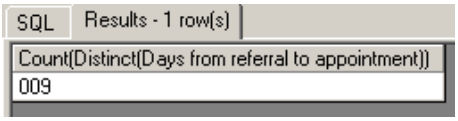
9.1.2 Count(Distinct())

The **Count(Distinct())** function counts the number of "distinct" (unique) values recorded in the field chosen. It behaves like **Count()** but eliminates duplicate values before the count is calculated.

This example produces a count of the number of different values have been recorded in the **Days from Referral to Appointment** field.



One row is returned:

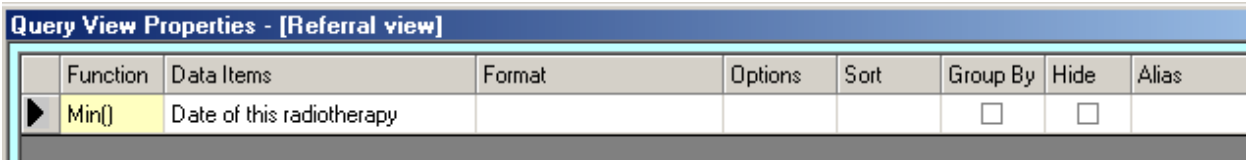


Note that Count(Distinct()) should normally return a lower value than the Count() function since (Count) simply returns how many records have a value entered where as Count(Distinct()) returns the number of different values entered.

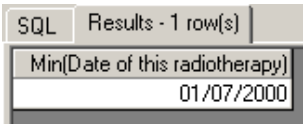
Count(Distinct) can only be used in SQLServer databases.

9.1.3 Min()

The **Min()** function finds the lowest or earliest value. This example returns the earliest **Date of Radiotherapy** in records that meet the filter criteria.

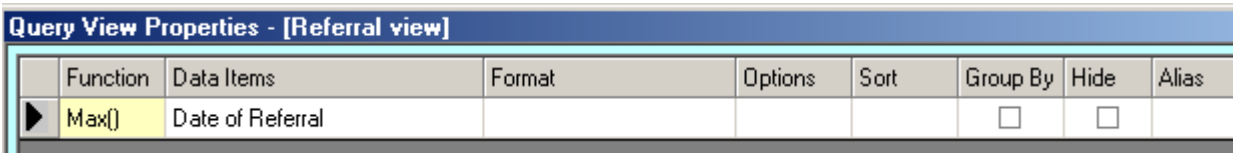


One record is returned:

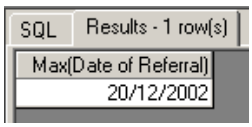


9.1.4 Max()

The **Max()** function finds the highest or latest value. This example returns the latest **Date of Referral** in records that meet the filter criteria.

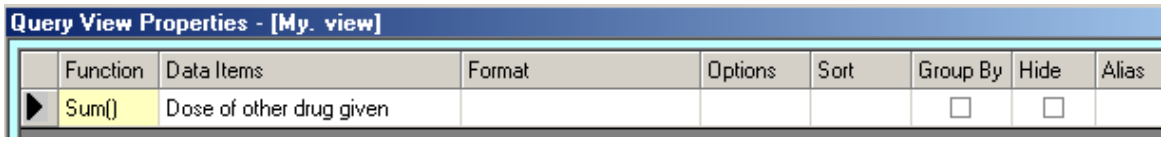


One record is returned:

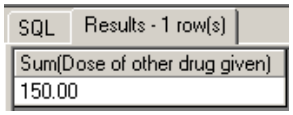


9.1.5 Sum

The **Sum()** function calculates the sum of the values that meet the filter criteria. This example returns the sum of all the Doses in the records that meet the filter criteria.

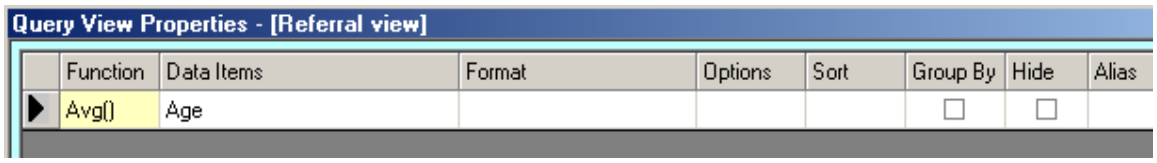


One row is returned:

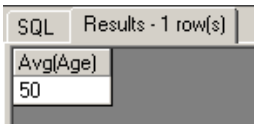


9.1.6 Avg()

The **Avg()** function finds the average of the values that meet the filter criteria. This example returns the average **Age** in records that meet the filter criteria.

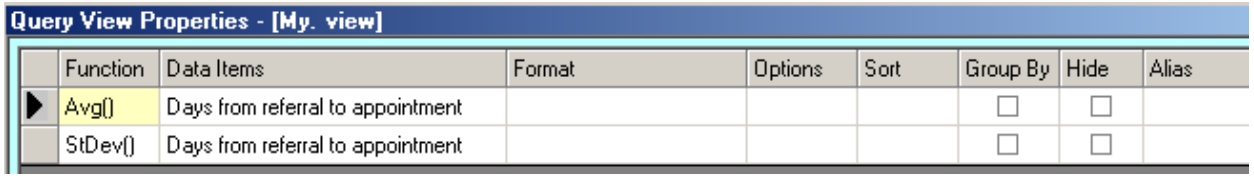


One record is returned:

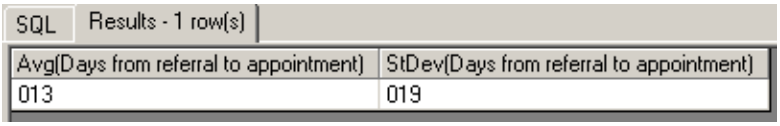


9.1.7 StDev()

The **StDev()** function finds the standard deviation of the values that meet the filter criteria. This example returns the standard deviation of **Age** in records that meet the filter criteria, along with the average **Age**.



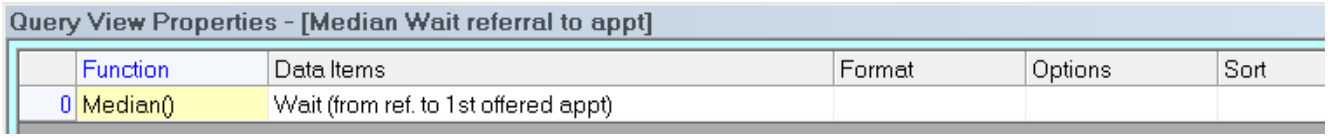
One row is returned:



9.1.8 Median()

The Median function should be used on numbers and returns the median value of the set of numbers.

For example:



The median function does not include missing and unknown values in its calculation, and ignores empty data.

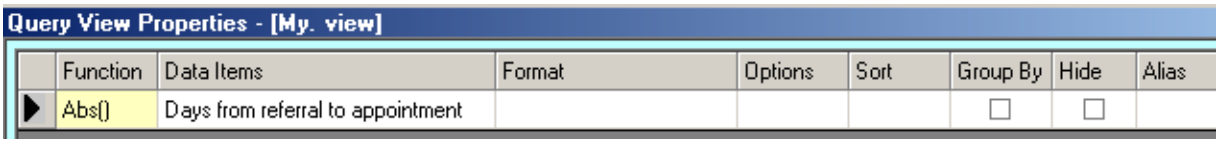
Important Note

The median function requires the database to be on Sql Server version 2005 or greater. The server on which the database resides will require Microsoft .Net 2.0 Framework to be installed. The median function requires a sql assembly and function to be registered, and this is done in a database update. The Median function is not available on Access databases or on SQL Server version 2000 databases.

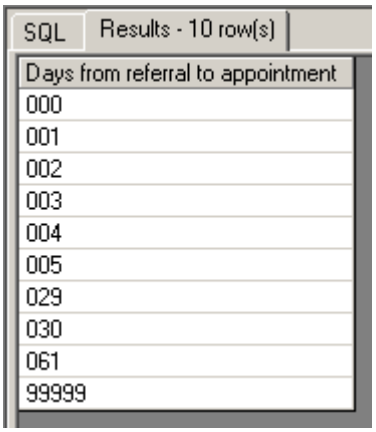
9.1.9 Abs()

The **Abs()** function returns the absolute positive value. Negative numbers are returned as positive numbers.

This example returns all the values in the **Days from Referral to Appointment** as positive numbers.



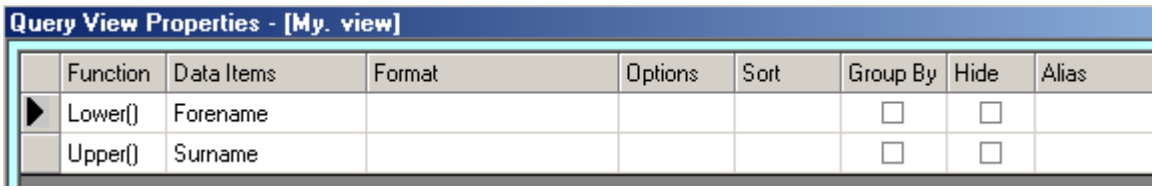
Note that even missing (-88888) and unknown (-99999) values are returned as positive.



9.1.10 Upper(), Lower()

The **Upper()** and **Lower()** functions set the case of the values to Upper or Lower.

This example



Query View Properties - [My. view]

	Function	Data Items	Format	Options	Sort	Group By	Hide	Alias
▶	Lower()	Forename				☐	☐	
	Upper()	Surname				☐	☐	

sets the **Forename** to lower case and the **Surname** to upper case.

SQL | Results - 19 row(s)

Lower(Forename)	Upper(Surname)
barry	BARTLEY
bill	BATES
bill	BROWN
chris	CLARKE
diane	DAVIES
francis	FORBES
gardenia	GATE
george	GRIFFITHS
greta	GREEN
helen	HUGHES
jessica	JOHNSTON
john	JONES
john	JOSEPH

9.1.11 Ltrim, Rtrim

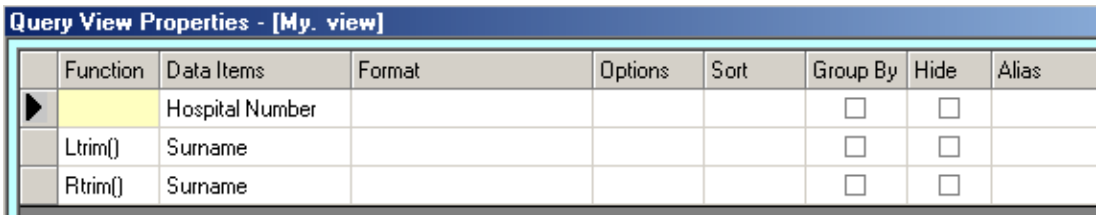
The **Trim** functions remove spaces (trim) to the left **(Ltrim)** or to the right **(Rtrim)** of the text.

In this example there are spaces leading and trailing some of the surnames. These results show how the surname is returned **before** the **trim** functions are applied:

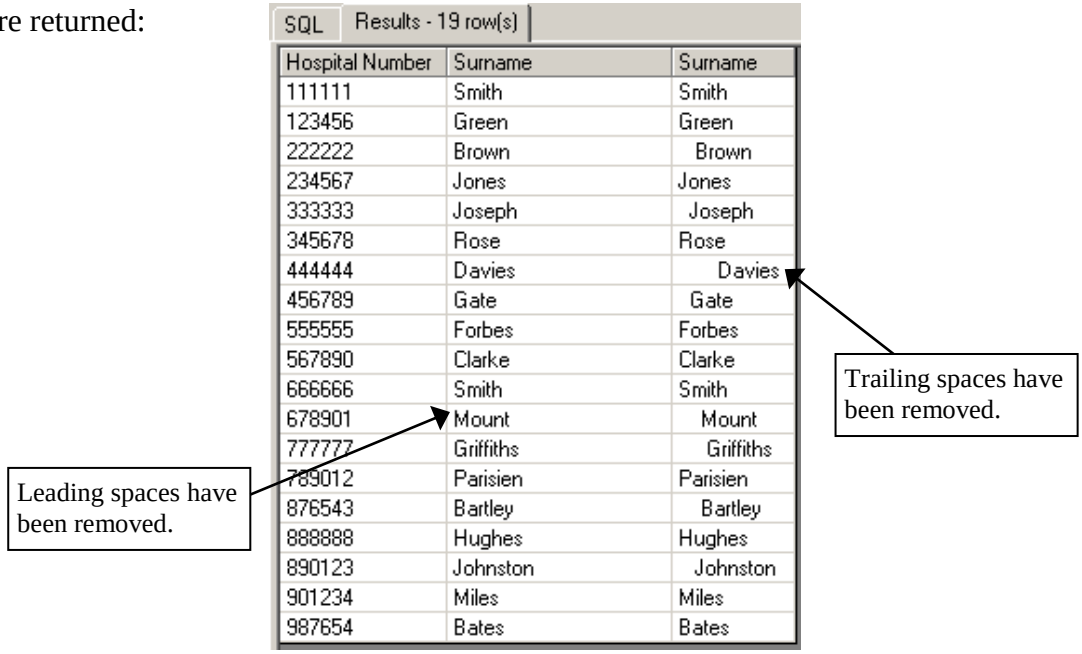
SQL | Results - 19 row(s)

Hospital Number	Surname	Surname
111111	Smith	Smith
123456	Green	Green
222222	Brown	Brown
234567	Jones	Jones
333333	Joseph	Joseph
345678	Rose	Rose
444444	Davies	Davies
456789	Gate	Gate
555555	Forbes	Forbes
567890	Clarke	Clarke
666666	Smith	Smith
678901	Mount	Mount
777777	Griffiths	Griffiths
789012	Parisien	Parisien
876543	Bartley	Bartley
888888	Hughes	Hughes
890123	Johnston	Johnston
901234	Miles	Miles
987654	Bates	Bates

This view removes **leading** spaces (**Ltrim**) from the surnames in the first column and removes **trailing** spaces (**Rtrim**) from the surnames in the second column:

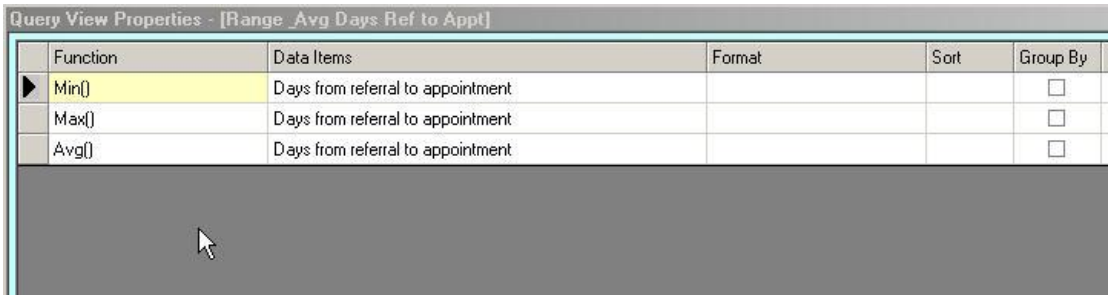


These results are returned:

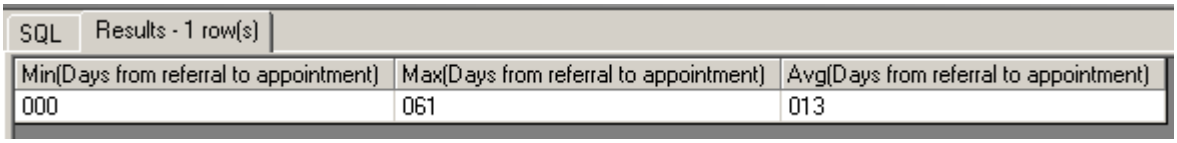


9.2 Multiple functions

Several functions can be used in the same view. This example finds the Min, Max and Average values of the **Days from Referral to Appointment** in records that meet the filter criteria.



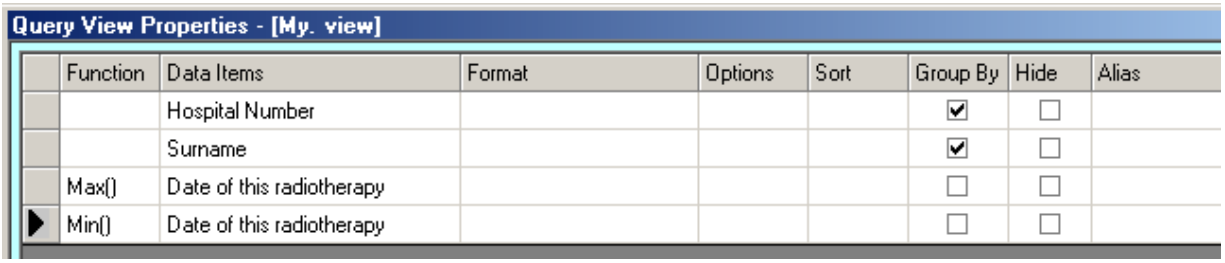
One row is returned:



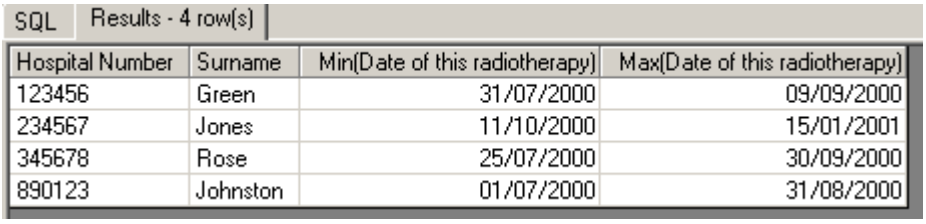
9.3 Aggregated values grouped by patient

Functions can also be used to find the aggregated values by patient. Any data items that are not being aggregated should have a tick in the **Group By** column.

This example returns the first and last date of radiotherapy for each patient meeting the filter criteria:



One row is returned for each patient that meets the filter criteria.



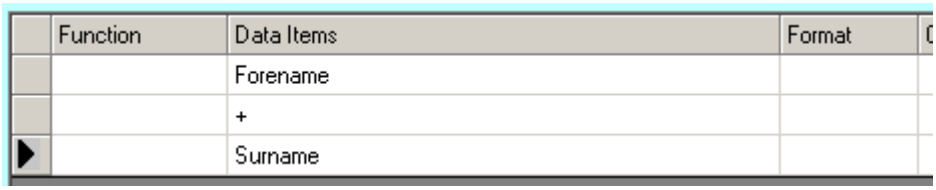
9.4 Concatenation

It is possible to concatenate two strings in a view.

The + symbol should be used in the syntax for SQLServer databases, and the & symbol for Access databases.

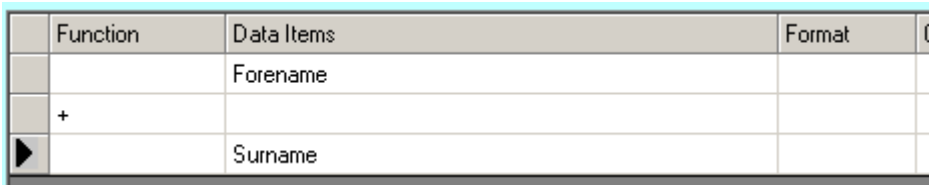
You can concatenate in the grid by adding the concatenation symbol as a value or an expression between the two items to be concatenated. When using a value, the concatenation symbol (and any additional string) should be entered in the **Data Items** column. When using an expression, the concatenation symbol (and any additional string) should be entered in the **Function** column.

In this example, a value field has been added between the two items.



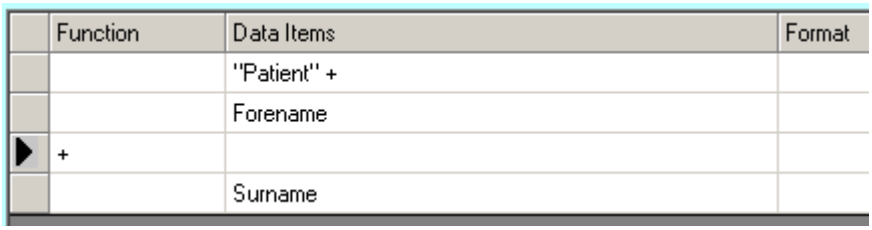
	Function	Data Items	Format
		Forename	
		+	
▶		Surname	

In this example, an expression field has been added between the two items.



	Function	Data Items	Format
		Forename	
	+		
▶		Surname	

You can also concatenate a constant with a data item. Strings to be concatenated must be contained within quotes.

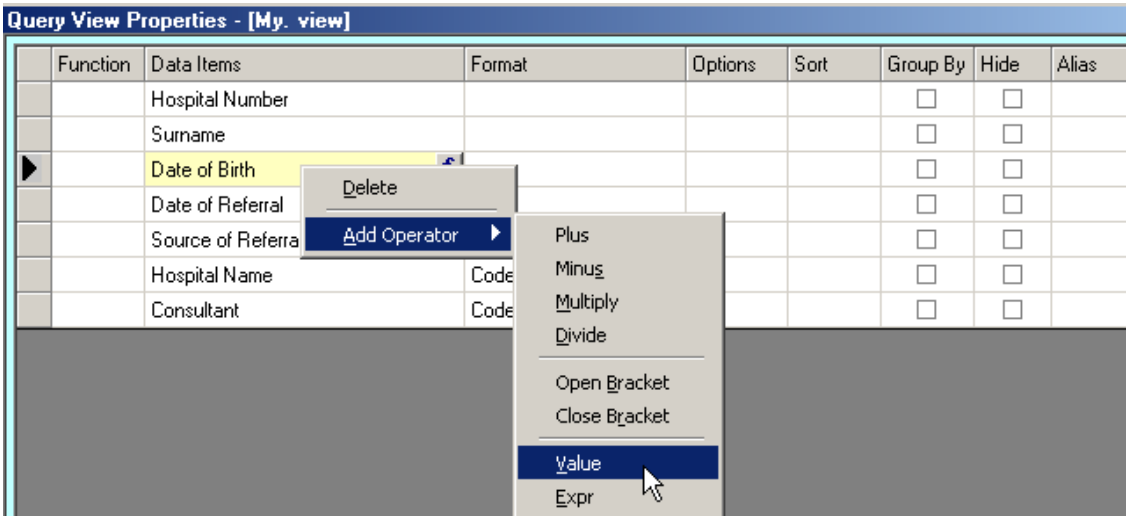


	Function	Data Items	Format
		"Patient" +	
		Forename	
▶	+		
		Surname	

9.5 Fixed Values in Views

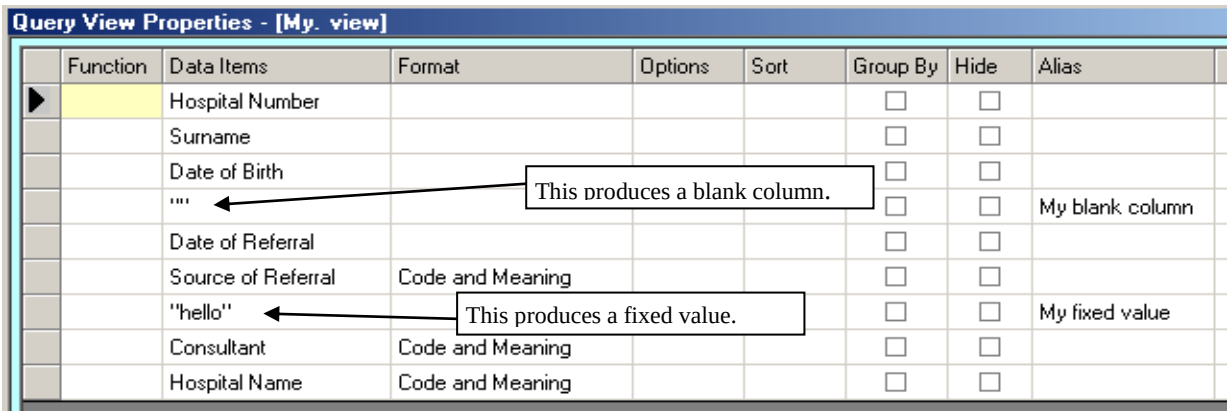
The **Value** operator can be used to insert a fixed value into a specific column or to insert a blank column. For example there might be a fixed record format for a data extract where a column should always have the same value or should always be blank.

To insert a fixed value into a view, right click the row above the fixed value and choose **Add Operator** then **Value**.

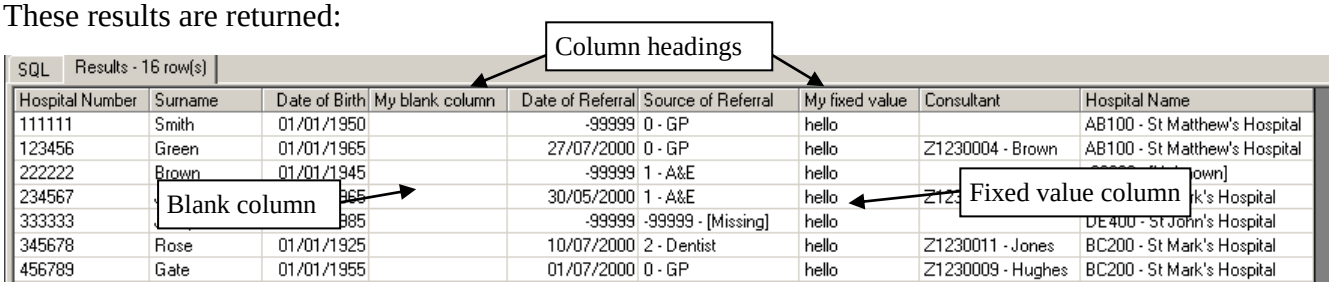


A row is inserted in the view. Type the fixed value directly into the **Data Items** column in the row. Note that you must put double quotes around the value.
To create a blank column in the view, type two double quotes in the **Data Items** column.

This example has two fixed values in it, one a blank column, the other with text. A column heading can be entered in the **Alias** column.



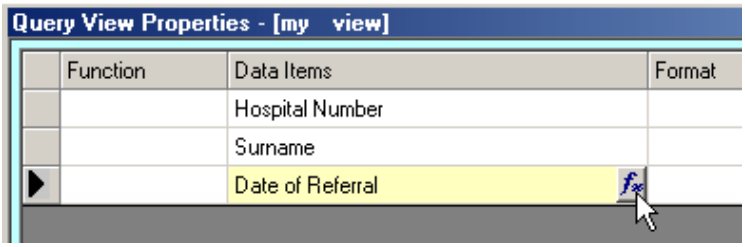
These results are returned:



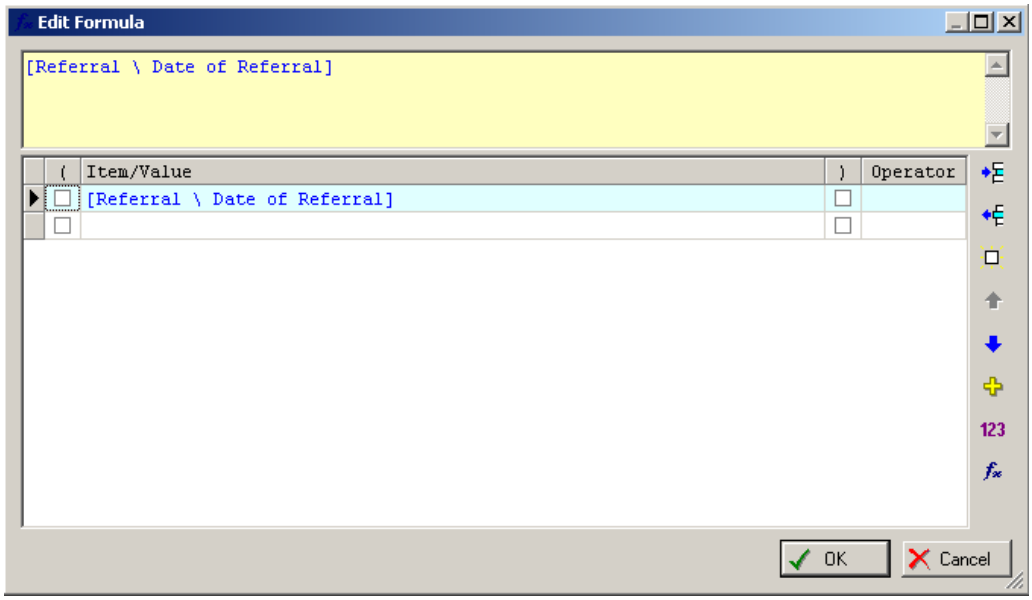
9.6 Expressions using the Formula Builder

Expressions can be added to views by using the **Formula builder.**

After adding an item to the view, press the ***fx*** button.



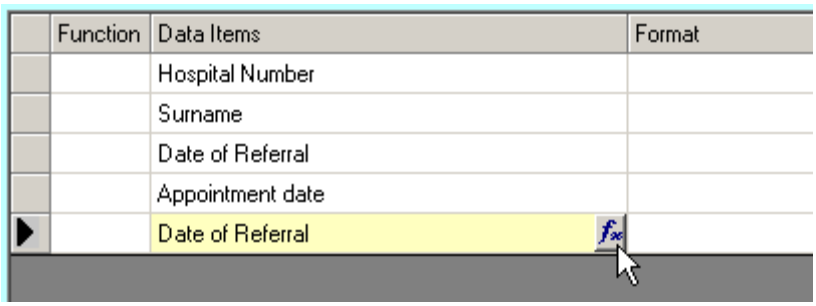
The **Edit Formula** dialog is displayed and you can create a formula in the usual way. Note that a smaller range of functions is available than is found in Design Management. This is because the expression must be turned into SQL and passed onto the database to evaluate, rather than being evaluated directly by InfoFlex.



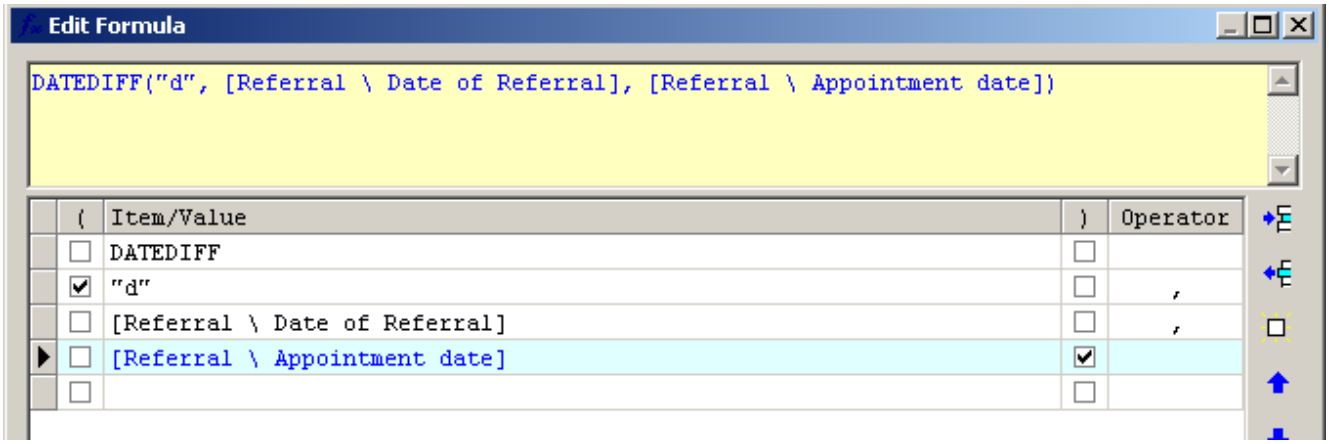
Example 1

This example uses the **DateDiff** function to calculate the difference between the Appointment Date and Date of Referral in the view

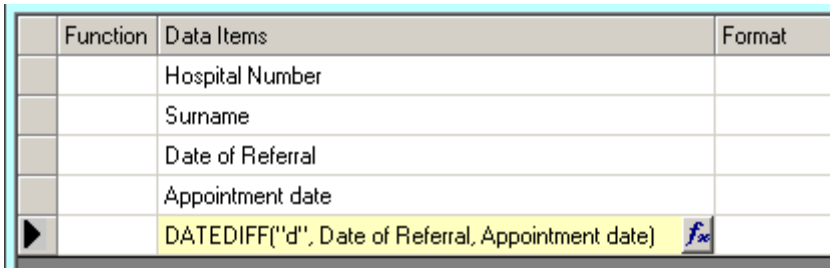
Add the **Date of Referral** item to the view and press the ***fx*** button.



Create the formula as shown below:



The formula is displayed in the view.



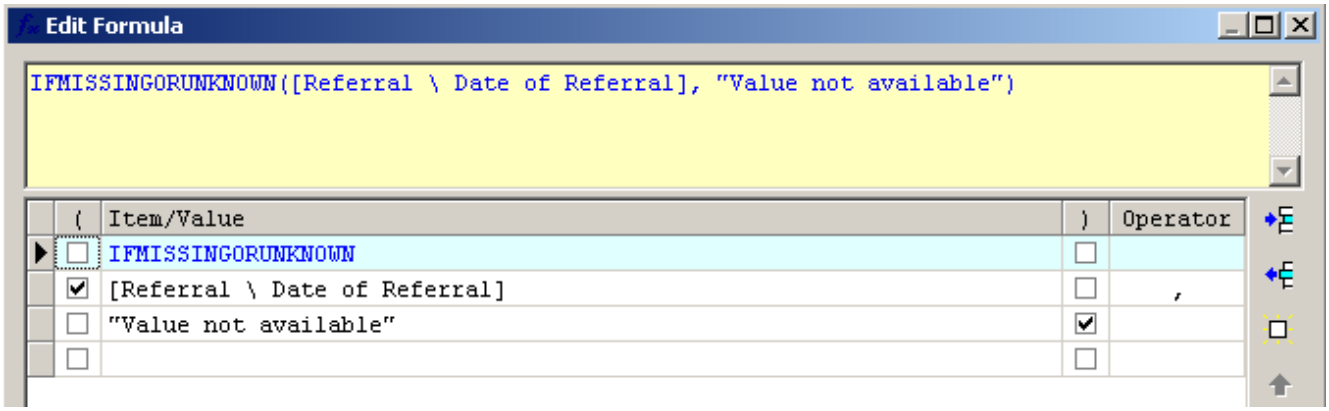
These results are returned:

SQL | Results - 20 row(s)

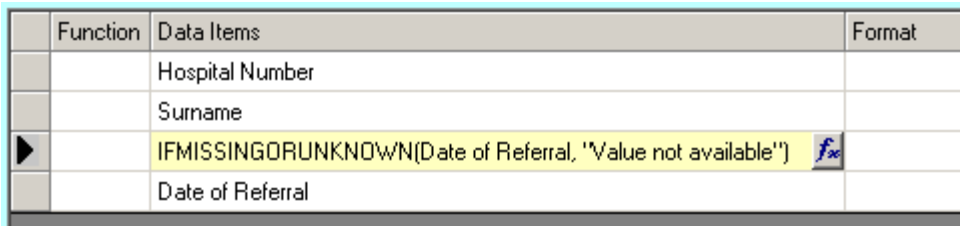
Hospital Number	Surname	Date of Referral	Appointment date	DATEDIFF('d',DateOfReferral,AppointmentDate)
111111	Smith			
666666	Smith	20/12/2002	20/01/2003	31
345678	Rose	10/07/2000	15/07/2000	5
789012	Parisien	02/04/2000	04/04/2000	2
678901	Mount	02/08/2000	02/08/2000	0
901234	Miles	01/01/2001	01/02/2001	31
901234	Miles	08/01/2000	12/01/2000	4

Example 2

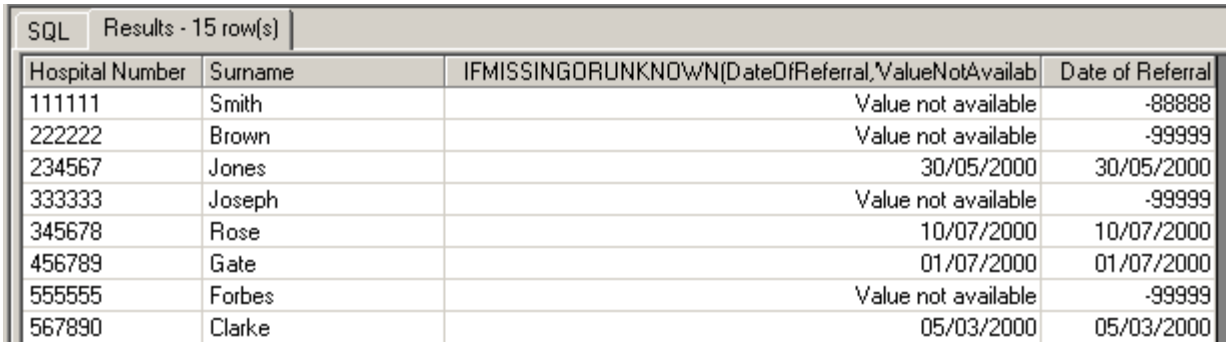
This example uses the **IFMISSINGORUNKNOWN** function to display free text if the Date of Referral is marked as missing (F11) or as unknown (F12).



The function is displayed in the view.



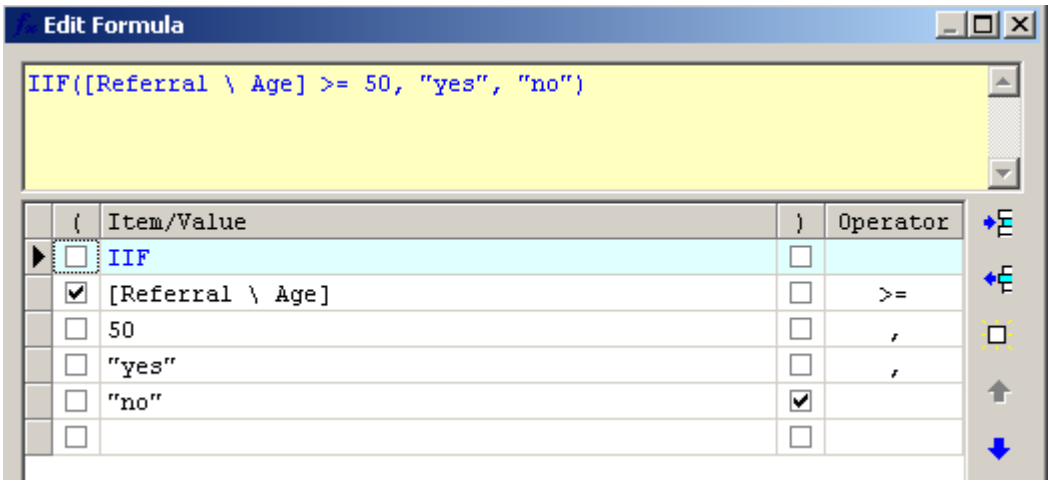
These results are returned (set **Outer Join** to **No** in the query definition):



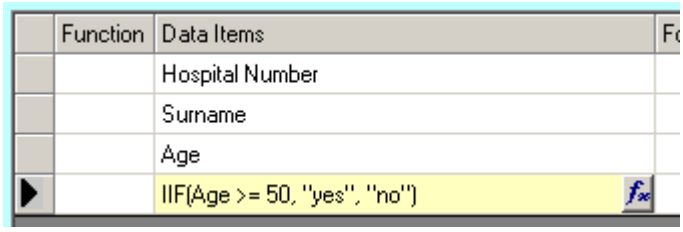
Hospital Number	Surname	IFMISSINGORUNKNOWN(DateOfReferral,'ValueNotAvailab	Date of Referral
111111	Smith	Value not available	-88888
222222	Brown	Value not available	-99999
234567	Jones	30/05/2000	30/05/2000
333333	Joseph	Value not available	-99999
345678	Rose	10/07/2000	10/07/2000
456789	Gate	01/07/2000	01/07/2000
555555	Forbes	Value not available	-99999
567890	Clarke	05/03/2000	05/03/2000

Example 3

This example uses **IIF** to display **yes** if the Age > 50 and **no** if it isn't.



The function is displayed in the view.



These results are returned:

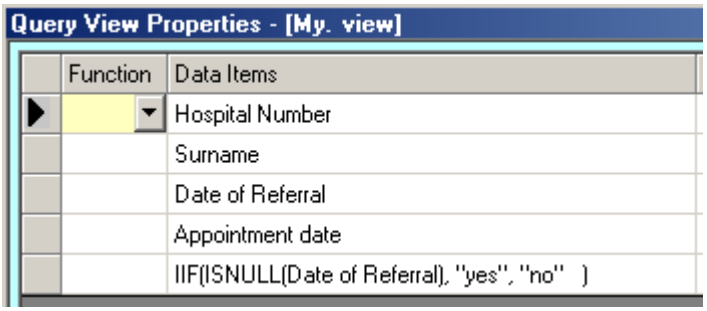
SQL | Results - 20 row(s)

Hospital Number	Surname	Age	IIF(Age>=50,'yes','no')
345678	Rose	76	yes
789012	Parisien	75	yes
987654	Bates	71	yes
890123	Johnston	65	yes
666666	Smith	63	yes
222222	Brown	58	yes
678901	Mount	56	yes
456789	Gate	46	no
567890	Clarke	40	no
876543	Bartley	40	no
123456	Green	36	no
234567	Jones	35	no
901234	Miles	21	no
901234	Miles	20	no
333333	Joseph	18	no
111111	Smith		no
444444	Davies		no

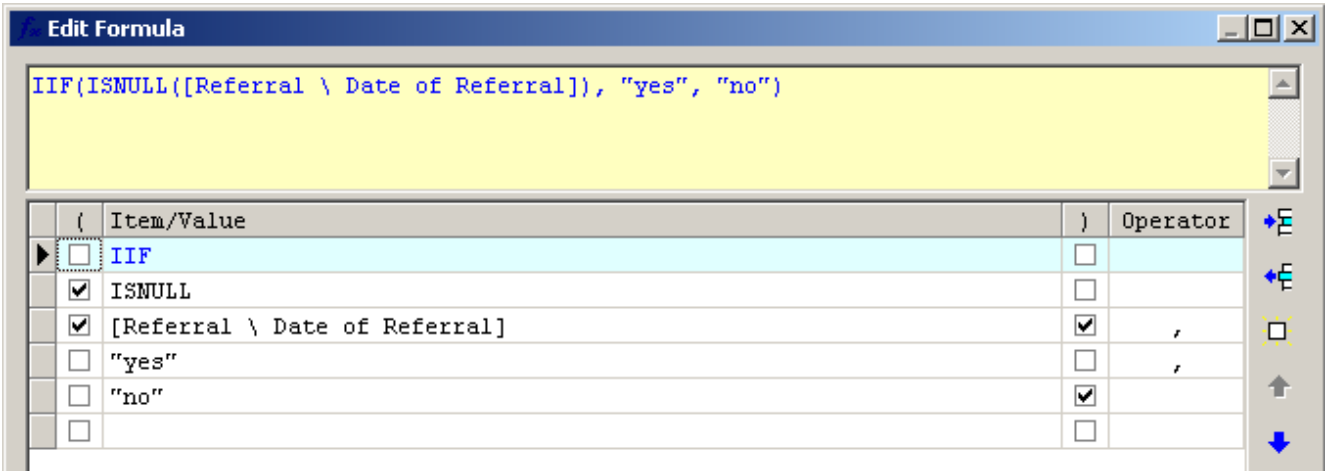
Example 4

The following example uses IIF displays **yes** if the Date of Referral is null and to display **no** if the Date of Referral is not null. Note that different syntax is required for Access and SQLServer for ascertaining if the Date of Referral is null.

The syntax for Access is as follows:



The above formula was created by adding the **Date of Referral** item to the view below the Appointment date item, then opening the formula builder and building the formula as shown below:



These results are returned:

SQL Results - 15 row(s)

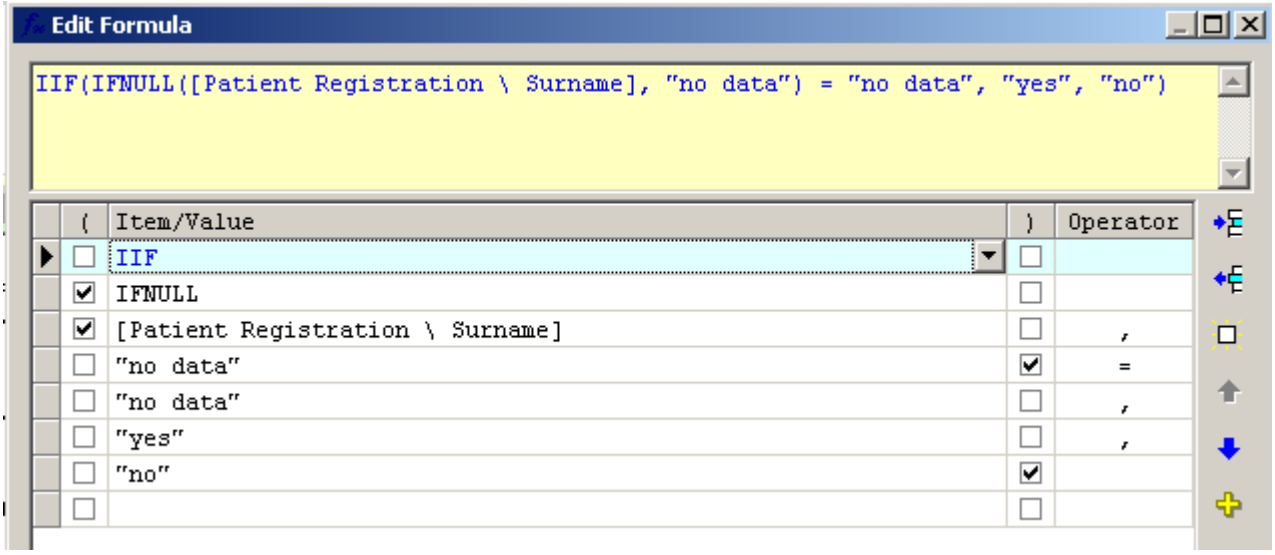
Hospital Number	Surname	Date of Referral	Appointment date	IIF(ISNULL(DateOfReferral),'yes','no')
111111	Smith	-88888	19/04/2010	no
666666	Smith		20/01/2003	yes
345678	Rose	10/07/2000	15/07/2000	no
789012	Parisien	02/04/2000	04/04/2000	no
678901	Mount		02/08/2000	yes
901234	Miles	08/01/2000	12/01/2000	no
333333	Joseph	-99999	20/01/2003	no

The syntax for SQLServer is as follows:

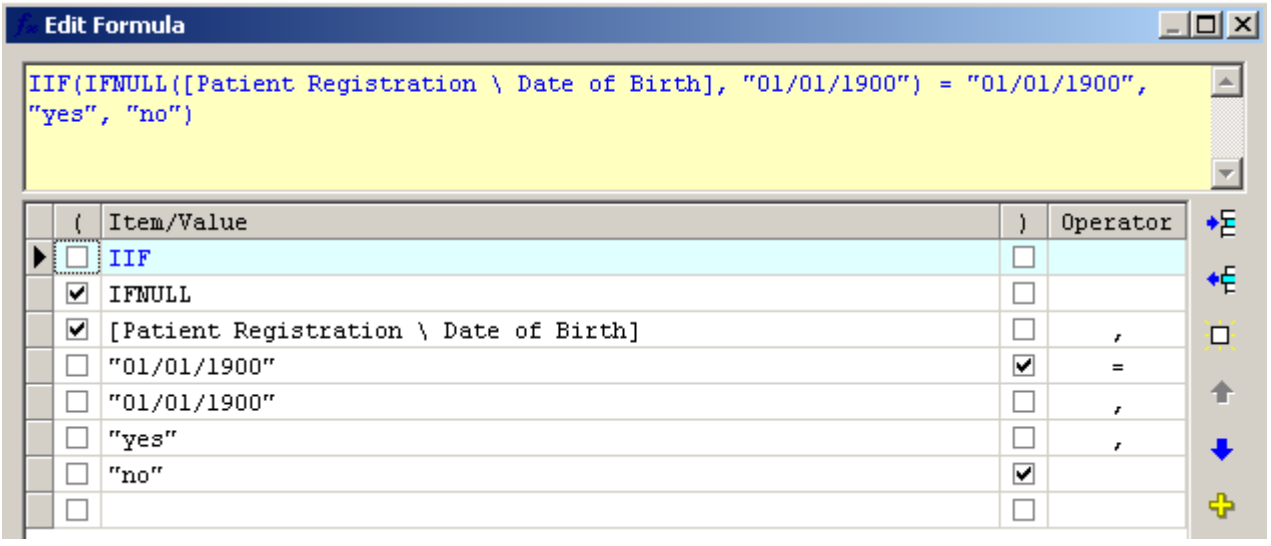
	Function	Data Items
		Hospital Number
		Surname
		IIF(IFNULL(Surname, "no data") = "no data", "yes", "no")
		Date of Birth
▶		IIF(IFNULL(Date of Birth, "01/01/1900") = "01/01/1900", "yes", "no")

Note that in SQLServer, the **IFNULL** statement is used to set an alternative value where the item is Null. (The alternative value set by the IFNULL statement must match the item type.)
The IIF statement is then used to set **yes** when the alternative value is present, and to set **no** otherwise.

The formula for the Surname item was created by adding the **Surname** item to the view then opening the formula builder and building the formula as shown below:



The formula for the Date of Birth item was created by adding the **Date of Birth** item to the view then opening the formula builder and building the formula as shown below:



These results are returned:

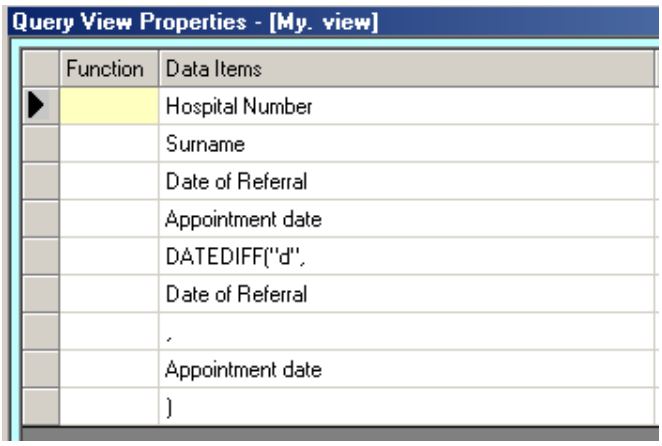
SQL | Results - 29 row(s)

Hospital Number	Surname	Yes if null	Date of Birth	Yes if null
123456		yes	01/01/1945	no
666666	Smith	no	01/01/1950	no
111111	Smith	no		yes
222222	Jones	no		yes
333333		yes	01/01/1982	no
444444	Jones	no		yes
555555		yes	01/01/1950	no
777777	Clarke	no	01/01/1922	no

9.6.1 Expressions without the formula builder

In earlier versions of InfoFlex before the formula builder was available in view definition, expressions could be added by using the **Value** operator to add the necessary syntax to the view. This method of calculating expressions is still supported and existing expressions created in this way can be edited. However wherever possible we recommend that expressions are created using the formula builder.

This example includes a calculation of the difference between the Appointment Date and Date of Referral using the DATEDIFF function.



The view was created using the following sequence:

- Double click **Hospital Number** item
- Double click **Surname** item
- Double click **Date of Referral** item
- Double click **Appointment Date** item
- Right click and select **Add Operator** and then select **Value**
- In the blank row enter **DATEDIFF("d",** in the **Data Items** column
- Double click the **Date of Referral** item.
- Right click and select **Add Operator** and then select **Value**
- In the blank row enter a comma in the **Data Items** column
- Double click the **Appointment date** item.
- Right click and select **Add Operator** and then select **Close Bracket**

These results are returned:

SQL | Results - 15 row(s)

Hospital Number	Surname	Date of Referral	Appointment date	DATEDIFF('d',DateOfReferral,AppointmentDate)
890123	Johnston	01/06/2000	06/06/2000	5
666666	Smith	20/12/2002	20/01/2003	31
234567	Jones	30/05/2000	01/07/2000	32
567890	Clarke	05/03/2000	05/05/2000	61

9.7 Simple Calculations in Views

Simple calculations can be carried out in views as described below.

Note: If calculations are regularly used in Data Analysis, it is worth considering creating the calculation in a calculated item in the design. This will simplify the view and improve the performance of the analysis.

Note: The function builder should be used to carry out calculations wherever possible.
In particular, date calculations should always be carried out using functions rather than simply subtracting one date from another. This is because you can choose the unit that the result is calculated in (days, months, hours etc). Also exceptions such as missing and unknown dates are handled better.

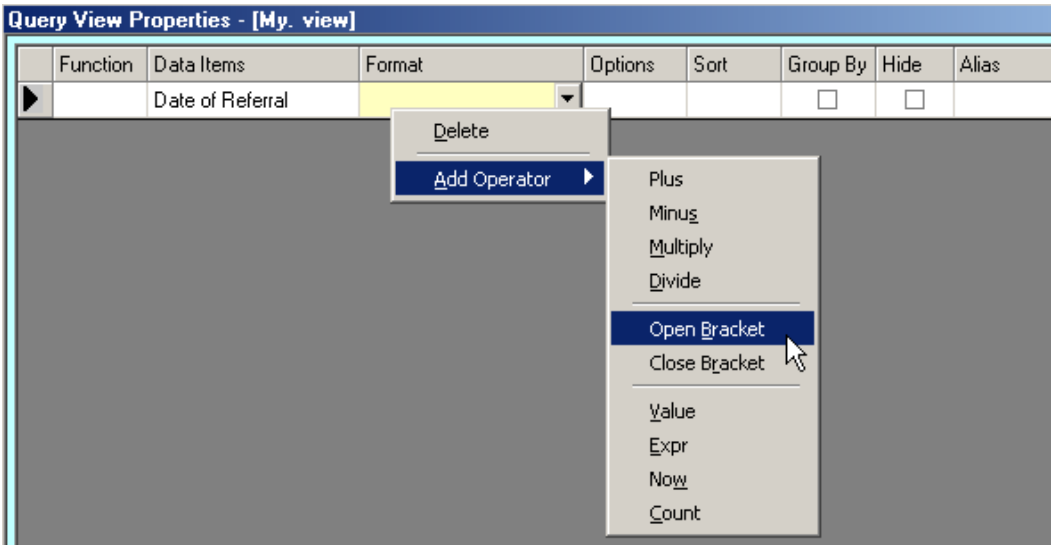
9.7.1 How to define calculations in views

If the view is the appropriate place for the calculation, the following operators can be used in the calculation: plus, minus, multiply, divide, brackets, **NOW** function, numeric values. Note that brackets should be used around the calculation to distinguish it from other view items.

It can be helpful to display the SQL on the **Views** tab (see section 5.3.3) whilst creating calculations in order to be sure that the syntax of the commas in functions etc is correct.

When a calculation is created in a view, the **Alias** column will need to be used to specify the column heading.

To add any of the above operators to a view, right click in the view and choose the operator you require.



Example 1

This example calculates a patient's weight loss by subtracting one weight from another. A column heading for the calculation has been entered in the **Alias** column.

	Function	Data Items	Format	Options	Sort	Group By	Hide	Alias
		Maximum weight				☐	☐	
		Current weight				☐	☐	
		(				☐	☐	
		Maximum weight				☐	☐	
		-				☐	☐	
		Current weight				☐	☐	
▶		)				☐	☐	Weight loss

These results are returned:

SQL | Results - 4 row(s)

Maximum weight	Current weight	Weight loss
100	085	15
092	090	2
080	063	17
120	097	23

Example 2

This example returns two columns. The first shows the **Date of Referral** and the second shows a calculation of the number of years between the **Date of Referral** and today's date. A column heading for the calculation has been entered in the **Alias** column.

Query View Properties - [My. view]

	Function	Data Items	Format	Options	Sort	Group By	Hide	Alias
▶		Date of Referral				☐	☐	
		(				☐	☐	
		NOW()				☐	☐	
		-				☐	☐	
		Date of Referral				☐	☐	
		)				☐	☐	
		/				☐	☐	
		365				☐	☐	No of years

These results are returned:

SQL | Results - 12 row(s)

Date of Referral	No of years
08/01/2000	10.2869275431253
05/03/2000	10.1307631595637
02/04/2000	10.0540508307965
30/05/2000	9.89514672120751
01/06/2000	9.88966726915271
01/07/2000	9.8074754883308
10/07/2000	9.78281795408422
27/07/2000	9.73624261161847

Note that this example can also be carried out using formula builder. Formula builder should be used wherever possible for calculations.

9.8 Summary of Operators that can be added to views

Below is a summary of the operators that can be added to views.

Plus	Adds a row containing **+**
Minus	Adds a row containing **-**
Multiply	Adds a row containing *****
Divide	Adds a row containing **/**
Open bracket	Adds a row containing **(**
Close bracket	Adds a row containing **)**
Value	Adds a blank row into which free text including numbers can be added
Expr	Adds a row and automatically displays the InfoFlex formula builder. For use with expressions that do not need InfoFlex items.
Now	Adds the **Now()** function
Count	Adds the **Count** operator (see section 8)

9.9 Exercises

9.9.1 Functions and multiple functions

The following exercises all use the Clinical Data view and items from the Referral event.

In the Clinical data view, create a new query group in the **Training** query group called **Functions exercises**.
Create a query within the group called **Functions**.
Create the views described below.
To test each view, select the **Functions** query and replace the view with the new view you have created. Run the query. There is no need for a filter.

1 Create a view which counts the number of **Appointment Dates**.

2 Create a view which counts the Distinct number of **Appointment Dates.**

3 Create a view which displays both the earliest **Date of Referral** and the latest **Date of Referral**.

4 Create a view which displays the sum of the **Duration of symptoms**.

5 Create a view which displays both the average **Days from Referral to Appointment** and the Standard deviation of **Days from Referral to Appointment**.

6 Create a view which displays the **Forename** in lower case and the **Surname** in upper case.

7 Create a view displays the lowest, highest and average **Days from Referral to Appointment**.

9.9.2 Aggregated values grouped by patient

This exercise creates a query returning the earliest and latest Date of Radiotherapy for each patient.

1 Create a view containing **Hospital Number** and **Surname** and two instances of **Date of this radiotherapy** (taken from the **Radiotherapy** event).

2 Tick the **Group By** column for **Hospital Number** and **Surname**.

3 Use the **Min** function for the first instance of **Date of this radiotherapy**.

4 Use the **Max** function for the second instance of **Date of this radiotherapy**.

5 Create a new query. Use the **All patients** filter and set **Outer Join** to **No**.

6 Run the query.

9.9.3 Concatenation

Create a view which concatenates **Forename** and **Surname** into one column.
(Note that Access databases require & and SQL databases require +)

Create a second view which concatenates **Forename** and **Surname** into one column and include a space between the Forename and Surname fields.

9.9.4 Fixed values

Make a copy of the **Referral** view and put it in the **Functions exercises** group.

Edit the copy of the **Referral** view.

Add a new blank column to the view. Name the column **My blank column**.

Add a second column which contains the word "Hello". Name the column **My fixed value**.

Make a copy of the **Referral** query and put it in the **Functions exercises** group.

In the copy of the Referral query, select the copy of the Referral view.

Run the copy of the **Referral** query and ensure the two columns are displayed.

9.9.5 Expressions in views

Recreate the examples in section 9.6. The examples are all taken from the Clinical data view and use data items from the **Patient registration** and **Referral** events.

Test the views in a query using the **All patients** filter.

9.9.6 Simple calculations

Recreate the examples in section 9.7. The examples are all taken from the Clinical data view and use data items from the **Patient registration** and **Referral** events. (The **Weight** items can be found in the Medical History panel of the **Referral** event).

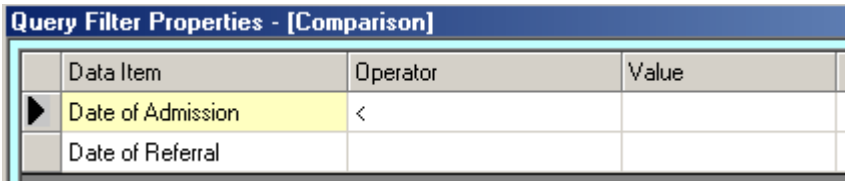
Test the views in a query using the **All patients** filter.

10 COMPARISONS AND CALCULATIONS IN FILTERS

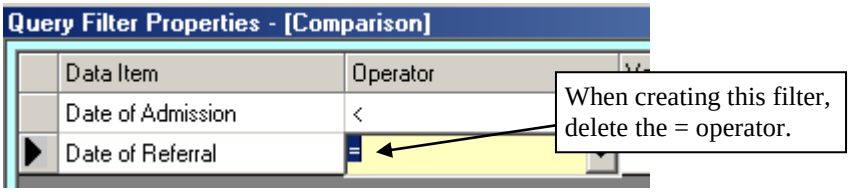
10.1 Comparison of Fields

Filters can be used to compare one field against another and thus return records where for example one date is earlier than another or one date or one field equals another field.

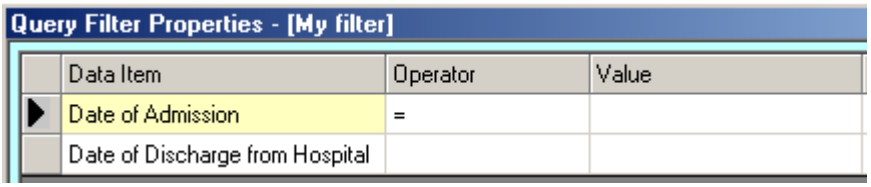
This example returns records where the **Date of Admission** is earlier than the **Date of Referral**.



Note that by default, when you add an item to a filter the ' = ' operator is always added. To create the above filter you will therefore need to remove the '= ' operator from the **Date of Admission** item.



This example returns fields where **Date of Admission** is the same as **Date of Discharge**.



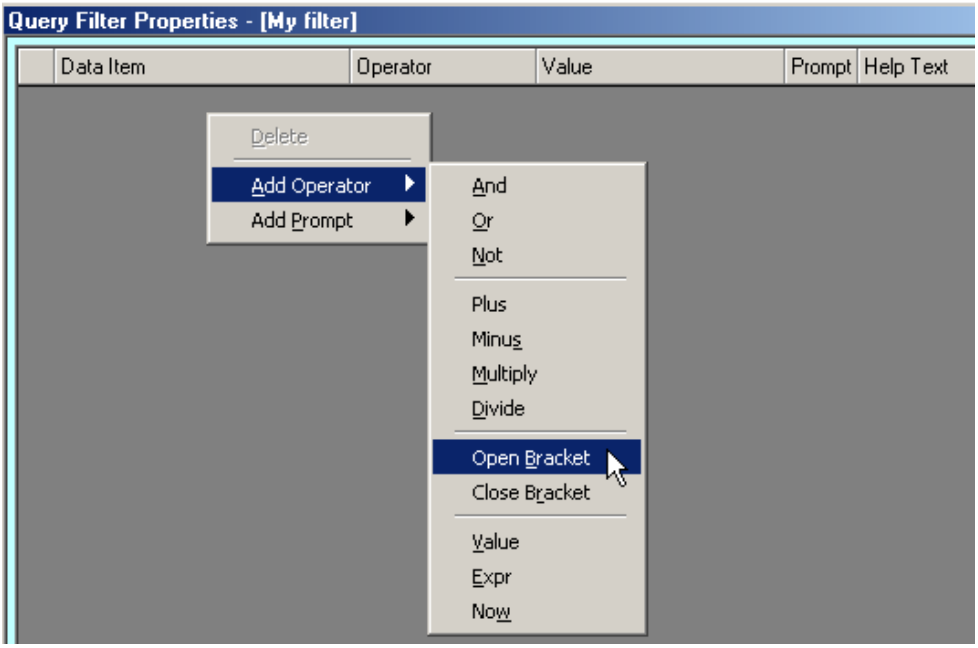
10.2 Calculations in Filters

Calculations can be carried out in filters. Plus, minus, multiply, divide, brackets and the **NOW** function can all be used in combination with data items. The **Value** and **Expression** operators can also be used to add values and expressions into the calculation.

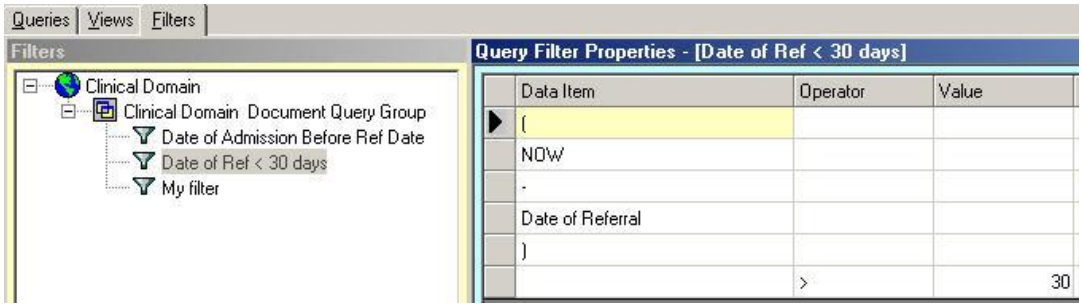
(Note that the **Expr** operator allows an operator to be entered in the **Operator** column whereas the **Value** operator does not. Both **Value** and **Expr** allow you to type free text in the **Value** column).

Note that it can be helpful to display the SQL on the **Filters** tab (see section 5.3.3) whilst creating calculations in order to be sure that the syntax and location of the commas in functions etc is correct.

To add any of the above operators to a filter, right click in the filter and choose the operator you require.



The following example returns records where **Date of Referral** is more than 30 days earlier than today's date.



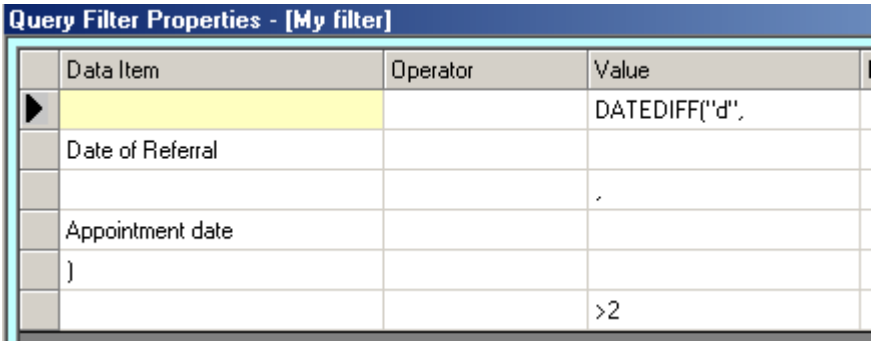
The above filter was created using the following sequence:

- Right click and select **Add Operator** and then select **Open Bracket**
- Right click and select **Add Operator** and then select **Now**
- Right click and select **Add Operator** and then select **Minus**
- Double click **Date of Referral** item and remove the = operator
- Right click and select **Add Operator** and then select **Close Bracket**
- Right click and select **Add Operator** and then select **Expr**
- In the blank row select > in the **Operator Column** and enter **30** in the **Value** column.

10.2.1 Adding functions to filters

The operators described above can be used to add the syntax of a function to a filter.

The following example returns records where the difference between Date of Referral and Appointment Date is greater than 2.



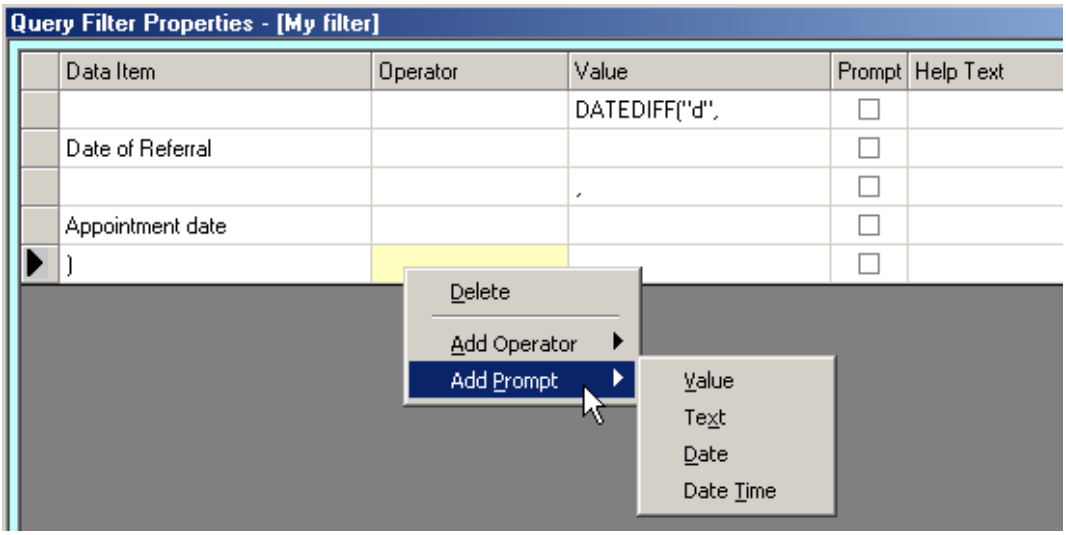
The above filter was created using the following sequence:

- Right click and select **Add Operator** and then select **Value.**
- Add **DATEDIFF("d",** to the **Value** column of the blank row.
- Double click the **Date of Referral** item and remove the = operator
- Right click and choose **Add Operator** then **Value**.
- Type a comma in the **Value** column of the blank row.
- Double click the **Appointment Date** item and remove the = operator
- Right click and select **Add Operator** then **Close Bracket**.
- Right click and select **Add Operator** then **Value**.
- Enter **>2** in the **Value** column of the blank row.

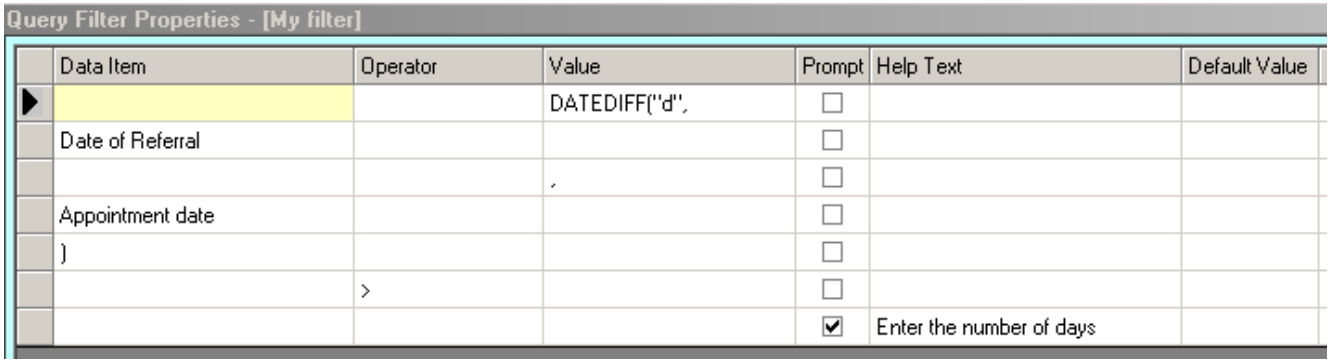
10.2.2 Add Prompt

When defining a calculation or function in a filter as described above, you can prompt for some of the elements instead of including them in the view.

To add a prompt for part of the calculation or function, right click in the filter and choose Add Prompt and then one of the available options.



The example in 10.2.1above can be adjusted to prompt for the number of days as follows:



The above filter was created using the following sequence:

- Right click and select **Add Operator** and then select **Value.**
- Add **DATEDIFF("d",** to the **Value** column of the blank row.
- Double click the **Date of Referral** item and remove the = operator
- Right click and choose **Add Operator** then **Value**.
- Enter a comma in the **Value** column of the blank row.
- Double click the **Appointment Date** item and remove the = operator
- Right click and select **Add Operator** then **Close Bracket**.
- Right click and select **Add Operator** then **Expr**.
- Enter > in the Operator column of the blank row.
- Right click and select **Add Prompt** then **Value**.
- Add the Help Text in the row with the Prompt column ticked.

10.3 Calculations in Filters using Fields containing Blanks

If a record has a null value in any of the items used in a filter calculation, that record will be not be returned in the query results.

If such a record does need to be included in the query results, then the null can be substituted with a default value such as a zero.

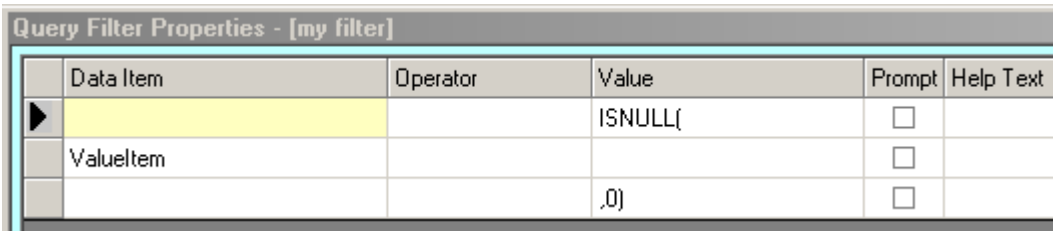
Access and **SQL Server** databases use different syntax for this.

SQL Server databases

The syntax to use **0** instead of **Null** for a data item called **ValueItem** is as follows:

IsNULL(ValueItem,0)

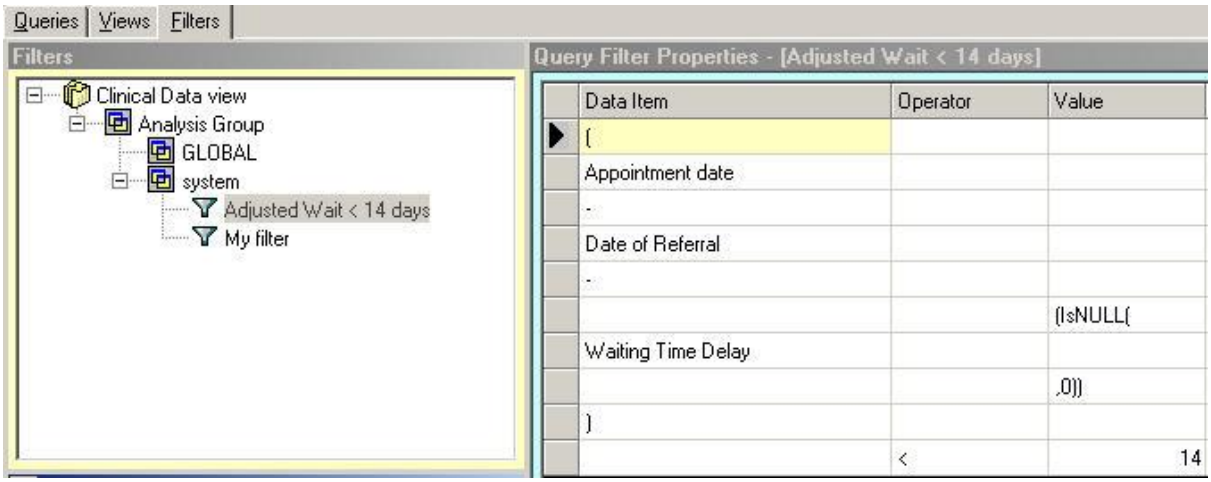
To use this syntax in a filter, insert a **Value** row before and after the data item in the filter, and type the relevant syntax in the Value column of the blank row.



The above filter was created using the following sequence:

- Right click and select **Add Operator** and then select **Value.**
- Add **ISNULL(** to the **Value** column of the blank row.
- Double click the **ValueItem** item and remove the = operator
- Right click and choose **Add Operator** then **Value**.
- Enter **,0)** in the **Value** column of the blank row.

Here is an example of the syntax in use in a filter:

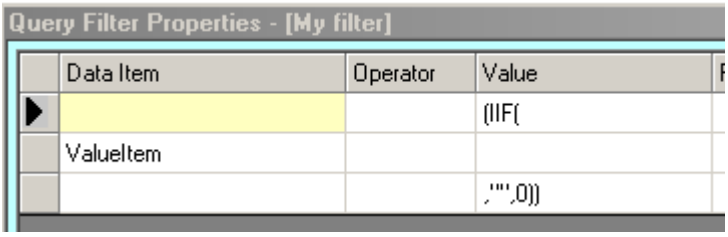


Access databases

The syntax to use **0** instead of **Null** for a data item called **ValueItem** is as follows:

(IIF(ValueItem,“”,0))

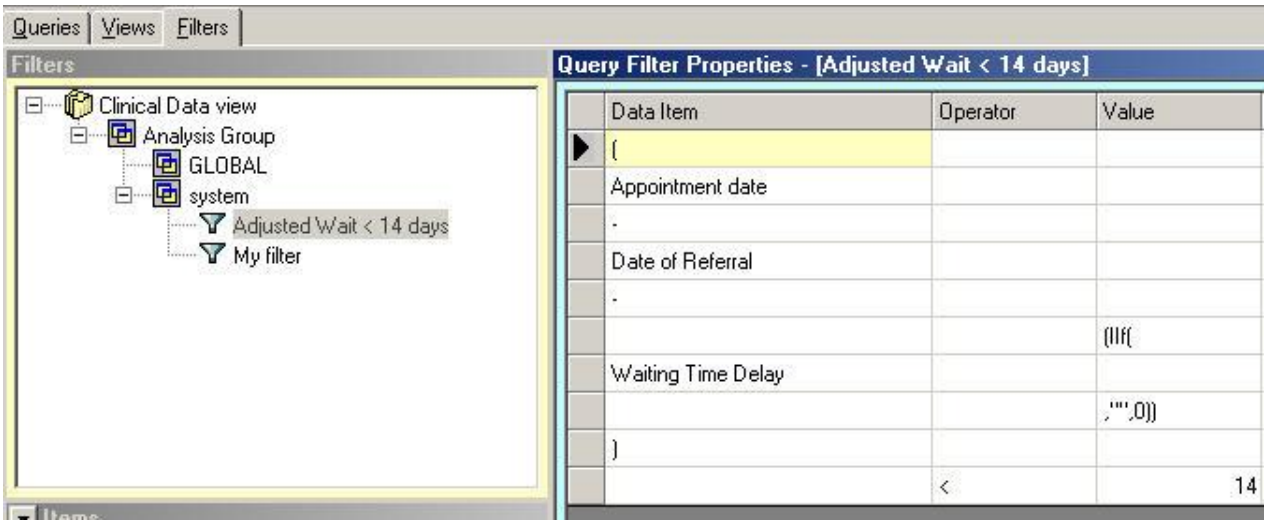
To use this syntax in a filter, insert a **Value** row before and after the data item in the filter, and type the relevant syntax in the Value column of the blank row.



The above filter was created using the following sequence:

- Right click and select **Add Operator** and then select **Value.**
- Type **(IIF(**in the **Value** column of the blank row.
- Double click the **ValueItem** item and remove the = operator
- Right click and choose **Add Operator** then **Value**.
- Enter **, “”,0))** in the **Value** column of the blank row.

Here is an example of the syntax in use in a filter:



10.4 Summary of Operators that can be added to filters

Below is a summary of the operators that can be added to views:

And	Adds a row containing **AND**
Or	Adds a row containing **OR**
Not	Adds a row containing **NOT**
Plus	Adds a row containing **+**
Minus	Adds a row containing **-**
Multiply	Adds a row containing *****
Divide	Adds a row containing **/**
Open bracket	Adds a row containing **(**
Close bracket	Adds a row containing **)**
Value	Adds a blank row into which numbers, letters or symbols can be entered in the **Value** column.
Expr	Adds a blank row into which numbers or letters can be entered in the **Value** column. An operator can be selected in the **Operator** column.
Now	Adds the **Now()** function

The following prompt operators can be added to filters:

Value	The user is prompted for a value to include in the calculation
Text	The user is prompted for text to include in the calculation
Date	The user is prompted for a date to include to the calculation
Date/time	The user is prompted for a date/time to include to the calculation

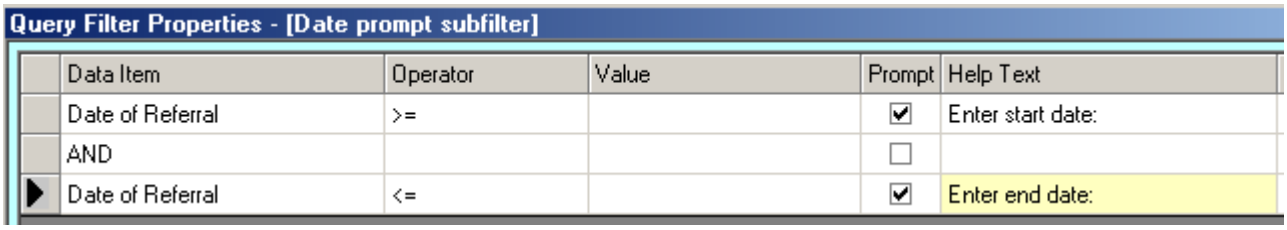
10.5 Using Subfilters in Filters

There can be common sub-elements in filters. Rather than create the common elements within each filter, a **subfilter** can be created that is then re-used in all the relevant filters. For example a time period using prompt parameters might be a common element of several filters. This prompted time period can be defined as an individual filter and then re-used in other filters as necessary.

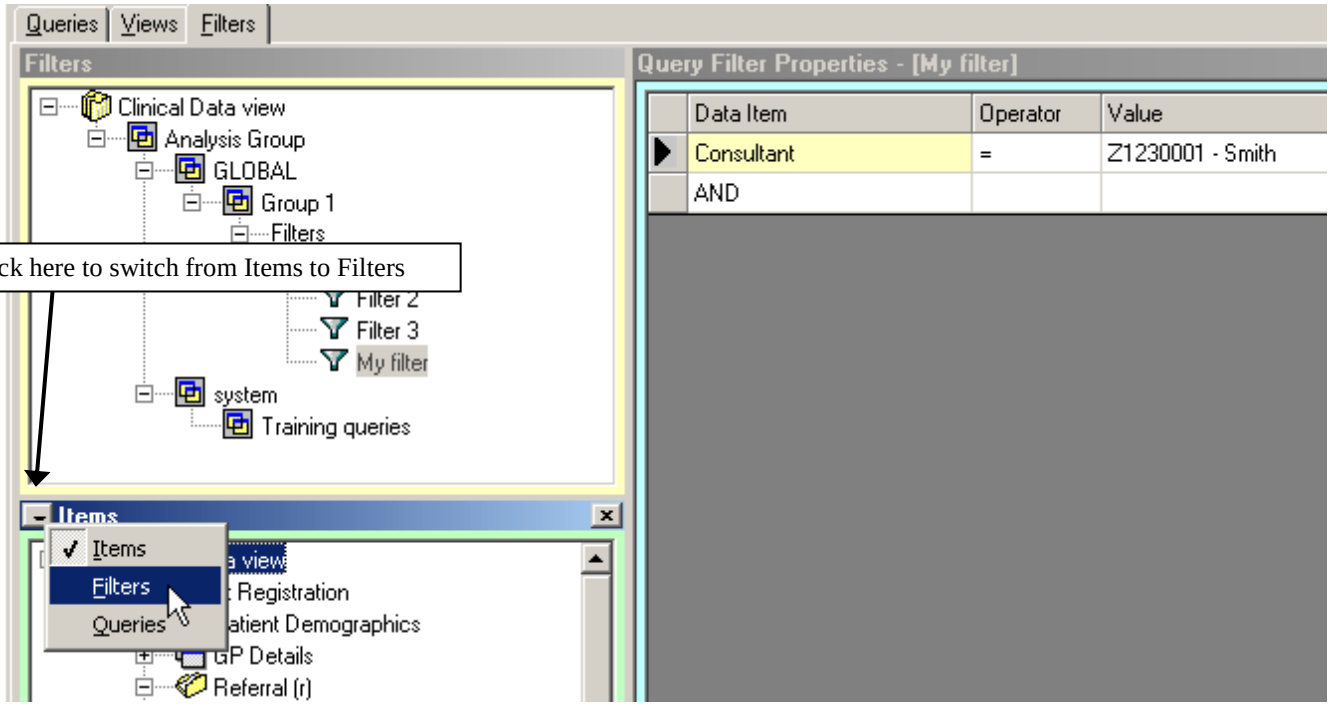
Creating a subfilter reduces maintenance since when changes are needed they can be made once to the subfilter rather than to every filter which uses the common element. However care must also be taken since changing a **subfilter** will globally affect all filters that use the subfilter.

We recommend using specific **Query groups** to identify subfilters.

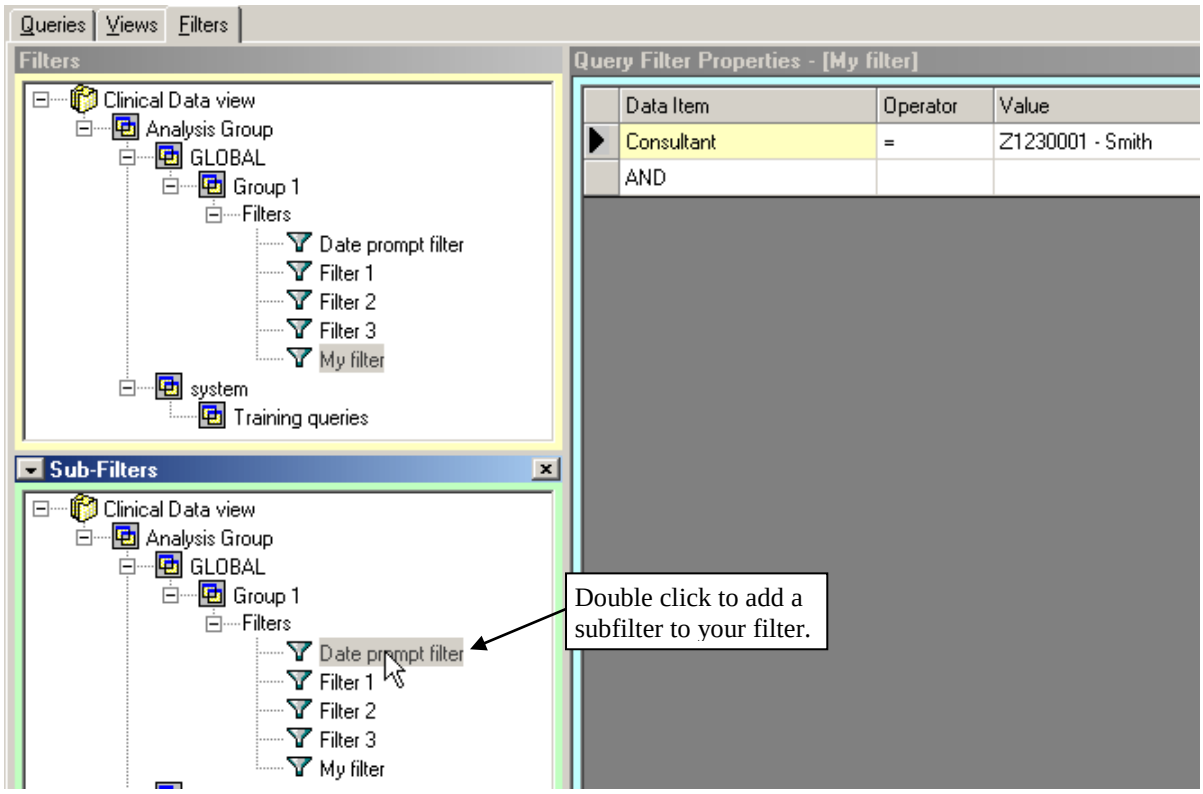
In the example below the filter called **Date prompt subfilter** prompts the user for start and end dates.



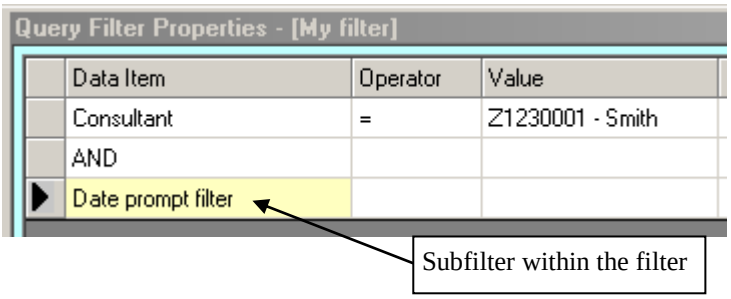
To use this filter as a subfilter in another filter, first create the second filter.
When you need to add the subfilter. switch the **Items** tree to display **Filters** rather than **Items**.



Filters are now available for selection. To add a filter to the filter, simply double click the filter you wish to add



The subfilter is displayed within the filter.



10.6 Using Subqueries in Filters

Filters can contain subqueries.

For example, a subquery might return a list of hospital numbers and the filter criteria specify that Hospital Number is in (or not in) the list of hospital numbers (in addition to other filter criteria) ie **" Hospital Number IN Subquery "** (where **Subquery** returns a list of hospital numbers).

Where subqueries are used in filters, the view in the subquery must contain only one item, and the filter criteria item be the same item that is used in the view of the subquery.

Example

A subquery has been defined with this filter:

Query Filter Properties - [subquery]

	Data Item	Operator	Value
▶	Presenting Symptoms	CONTAINS CODE	3 - nausea

The subquery returns three patients who meet the filter criteria:

SQL | Results - 3 row(s)

Hospital Number
456789
890123
987654

A filter is defined using the subquery:

	Data Item	Operator	Value
▶	Hospital Number	IN	
	Subquery		
	AND		
	Performance Status	=	0 - fully active

Filter criteria using the subquery. **Hospital number** is the item in the view of the subquery.

This filter will find patients who are in the list of patients returned by the subquery and who additionally have a performance status of **Fully active**.

Notice that filter criteria item used with the subquery is the same as the item used in the view of the subquery.

The query using the subfilter returns these results:

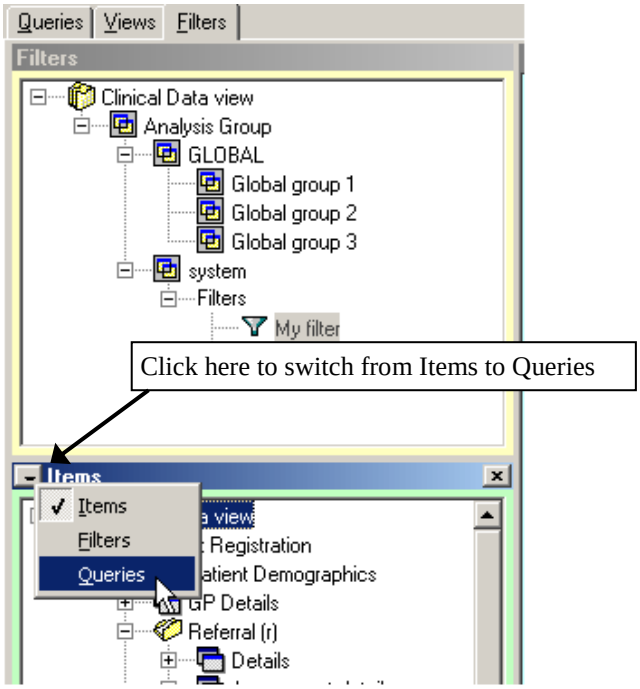
SQL | Results - 1 row(s)

Hospital Number	Presenting Symptoms	Performance Status
890123	3 - nausea; 4 - vomiting; 99 - Other	0 - fully active

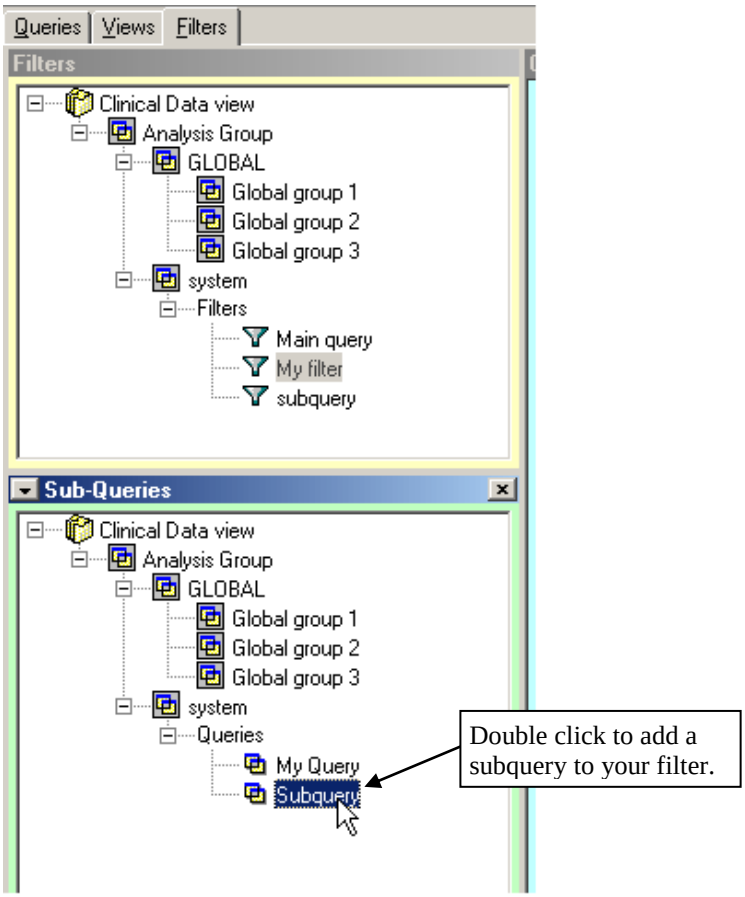
Only one patient is returned since only one of the patient in the results returned by the subquery also has a Performance Status of Fully Active.

10.6.1 Selecting a subquery in a filter

To use a query as a subquery in a filter, first create the query you wish to use as the subquery. Next create the filter. When you need to add the subquery, switch the **Items** tree to display **Queries** rather than **Items**.



Queries are now available for selection. To add a query to the filter, simply double click the query you wish to add.



10.7 Exercises

10.7.1 Comparison of fields

1 Create a filter in **Training group 2** to find records where the **Date of Admission** (**Referral** event, **Surgery details** panel) **<= Date of Referral**.

2 Make a copy of **Referral** view and put it in Training group 2. Edit the copy of the Referral view and add the **Date of admission** (**Referral** event, **Surgery details** panel) to the view.

3 Create a query in **Training group 2** using the copy of the **Referral** view and the filter you have just created. Run the query to test your filter.

10.7.2 Calculations in filters

Create a filter which finds records where the **Date of Referral** is more than 30 days earlier than today's date.

1 Create a filter in **Training group 2** with the calculation **(Now – Date of Referral) > 30**. (See section 10.2).

2 For testing, create a view in **Training group 2** which contains the **Date of Referral** and a calculation of **(Now - Date of referral)**.

3 Create a query in **Training group 2** using the above view and filter and check that the correct results are returned.

Create a filter in **Training group 2** which finds records where the difference between **Date of Referral** and **Appointment Date** is greater than 2.

1 Create a filter with the calculation **DateDiff("d", Date of Referral, Appointment Date) > 2**. (See section 10.2.1).

2 For testing, create a view in **Training group 2** which contains the **Date of Referral**, **Appointment Date** and the DateDiff calculation used in the filter.

3 Create a query in **Training group 2** using the above view and filter and check that the correct results are returned.

Create a filter in **Training group 2** which finds records where the difference between **Date of Referral** and **Appointment Date** is greater than a prompted value.

1 Edit the filter you created in the previous exercise so that the user is prompted for the greater than value. (See section 10.2.2).

2 Run the query you created in the previous exercise and check that the correct results are returned.

10.7.3 Subfilters

This exercise creates a filter that prompts for the earliest and latest **Date of Referral** and the **Consultant**. Since we have already created a filter that prompts for the earliest and latest Date of Referral, we shall use that filter as a subfilter.

1 Create a new query group called **Subfilters**.

2 Make a copy of the **Prompt filter** and paste it in the Subfilters query group.

3 Create a new filter in **Training group 2** called **Combined filter.**

4 Switch the Items list to display filters.
Add the copy of the **Prompt filter** in the **Subfilters** query group to the **Combined** filter.

5 Add the additional criteria **AND consultant = [prompt]** to the filter.

6 In **Training group 2,** create a view containing **Date of Referral** and **Consultant**.

7 In **Training group 2,** create a query using the view and filter you have just created. Run the query and check that the correct results are returned. (Test it with Date of Referral <= 01/01/2000 and >= 31/12/2000 and Consultant = Z1230002).

10.7.4 Subqueries

This exercise recreates the example in section 10.6 above.

1 Create a new query group called **Subqueries**.

2 In the **Subqueries** group, create a query called **My Subquery**. The view should contain the Hospital number only. The filter should specify **Presenting Symptoms CONTAINS CODE 3 – nausea**. Run the query and ensure three records are returned.

3 In **Training group 2**, create a filter called **Symptoms**. Set the criteria to
Hospital number IN My Subquery AND Performance status = 0 – Fully active.

4 In **Training group 2**, create a view called **Symptoms**. Include the Hospital Number, Presenting Symptoms and Performance Status.

5 In **Training group 2**, create a query called **Symptoms** and set the **Symptoms** view and the **Symptoms** filter. Run the query and ensure the records returned meet the filter criteria.

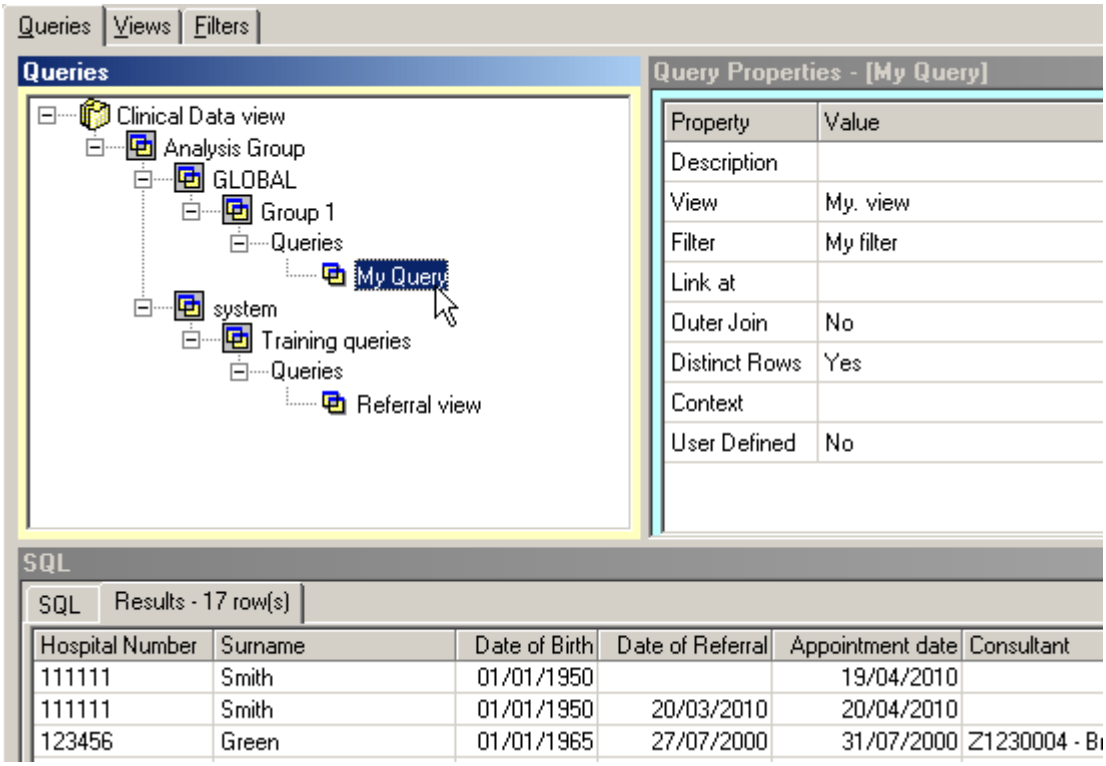
6 In the **Symptoms** filter, change **IN** to **NOT IN** and then run the **Symptoms** query. Ensure that the patients returned do not have symptoms of **3 -nausea** and are **Fully active**.

11 EXPORTING DATA

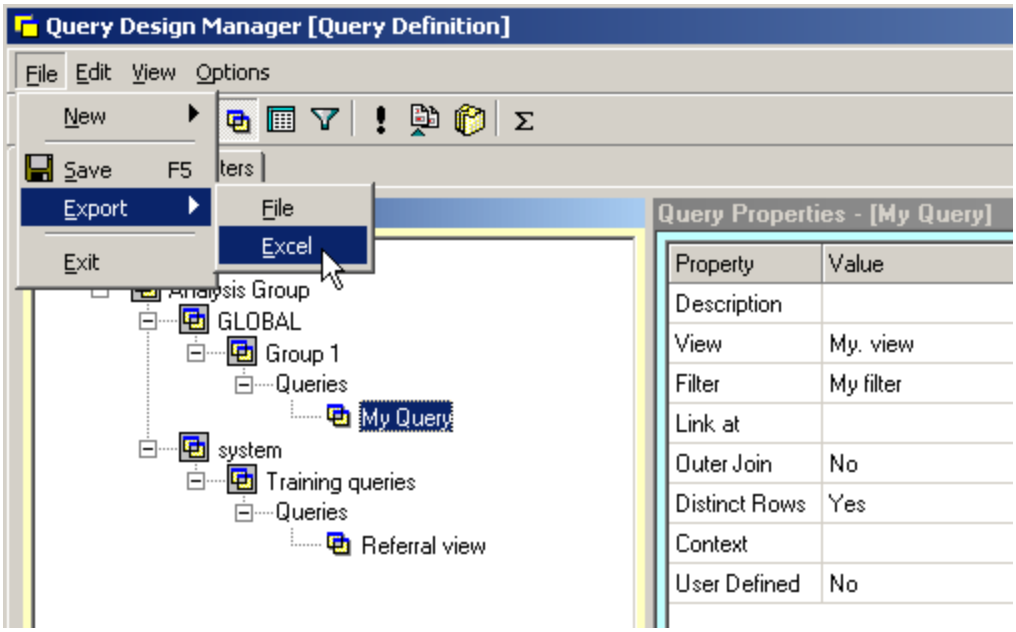
Data can be exported to a file or to Microsoft Excel directly from QDM.

11.1 Export to Microsoft Excel

To export your query results to MS Excel, select your query on the queries tab in **QDM**. If you wish, run a preview of it.

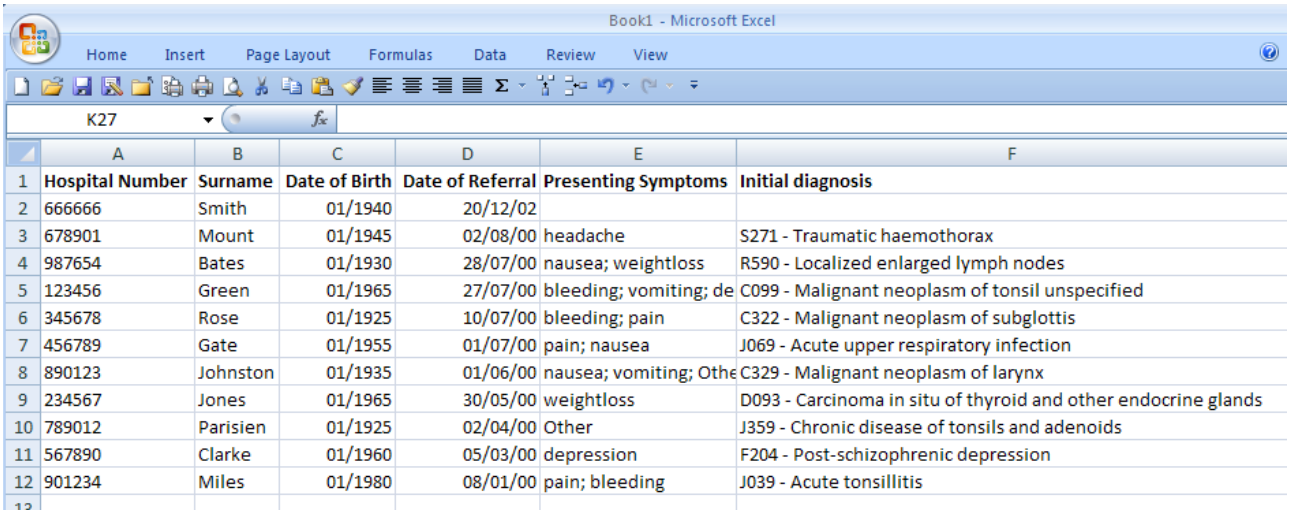


Go to the **File** menu and choose **Export** then **Excel**.



InfoFlex opens **MS Excel** and displays the query results in a new unsaved worksheet.

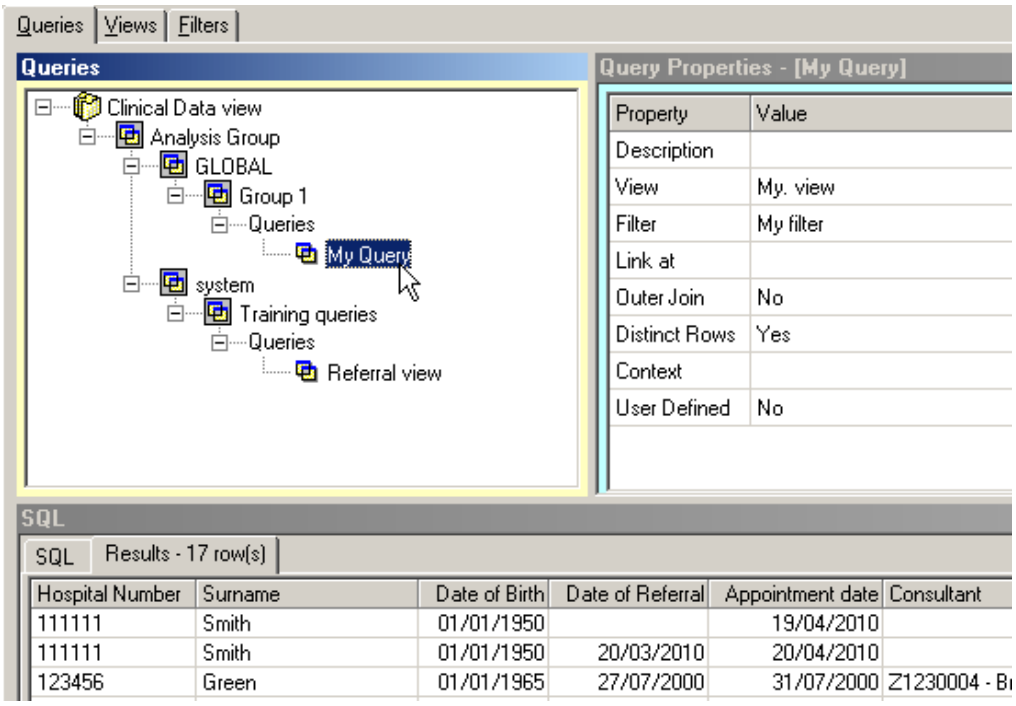
If you wish to keep this file to use in the future, you should save it in the format and location of your choice.



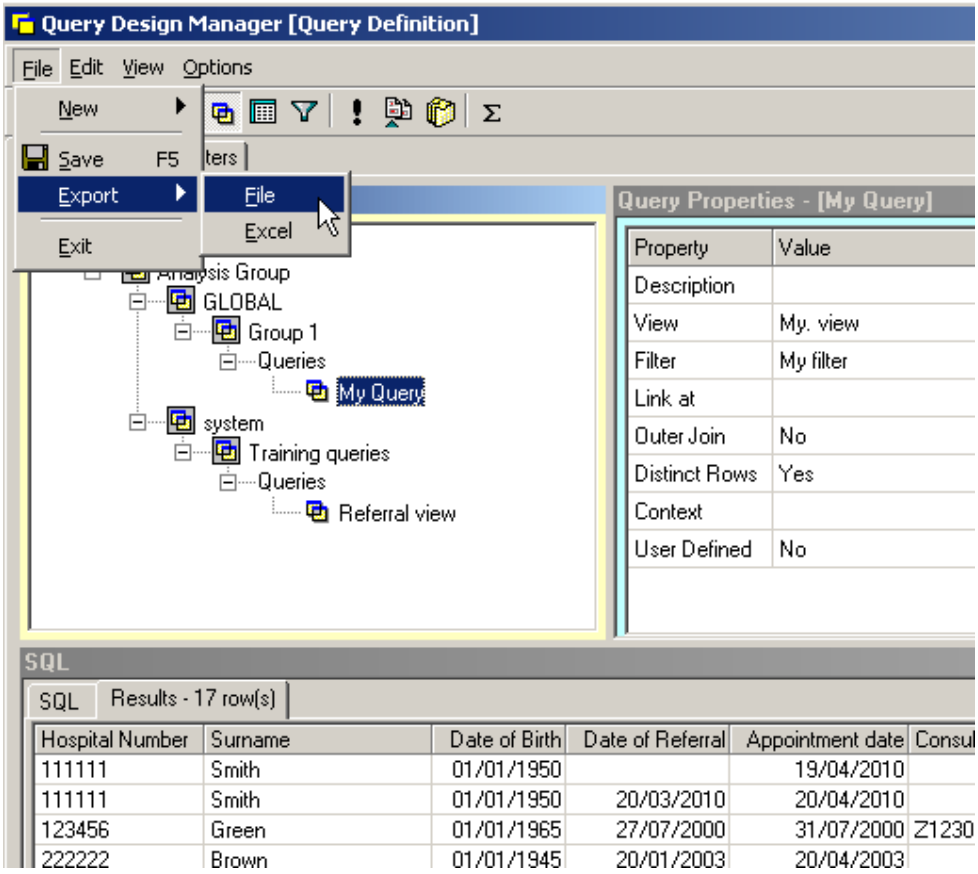
	A	B	C	D	E	F
1	**Hospital Number**	**Surname**	**Date of Birth**	**Date of Referral**	**Presenting Symptoms**	**Initial diagnosis**
2	666666	Smith	01/1940	20/12/02		
3	678901	Mount	01/1945	02/08/00	headache	S271 - Traumatic haemothorax
4	987654	Bates	01/1930	28/07/00	nausea; weightloss	R590 - Localized enlarged lymph nodes
5	123456	Green	01/1965	27/07/00	bleeding; vomiting; de	C099 - Malignant neoplasm of tonsil unspecified
6	345678	Rose	01/1925	10/07/00	bleeding; pain	C322 - Malignant neoplasm of subglottis
7	456789	Gate	01/1955	01/07/00	pain; nausea	J069 - Acute upper respiratory infection
8	890123	Johnston	01/1935	01/06/00	nausea; vomiting; Othe	C329 - Malignant neoplasm of larynx
9	234567	Jones	01/1965	30/05/00	weightloss	D093 - Carcinoma in situ of thyroid and other endocrine glands
10	789012	Parisien	01/1925	02/04/00	Other	J359 - Chronic disease of tonsils and adenoids
11	567890	Clarke	01/1960	05/03/00	depression	F204 - Post-schizophrenic depression
12	901234	Miles	01/1980	08/01/00	pain; bleeding	J039 - Acute tonsillitis

11.2 Export to File

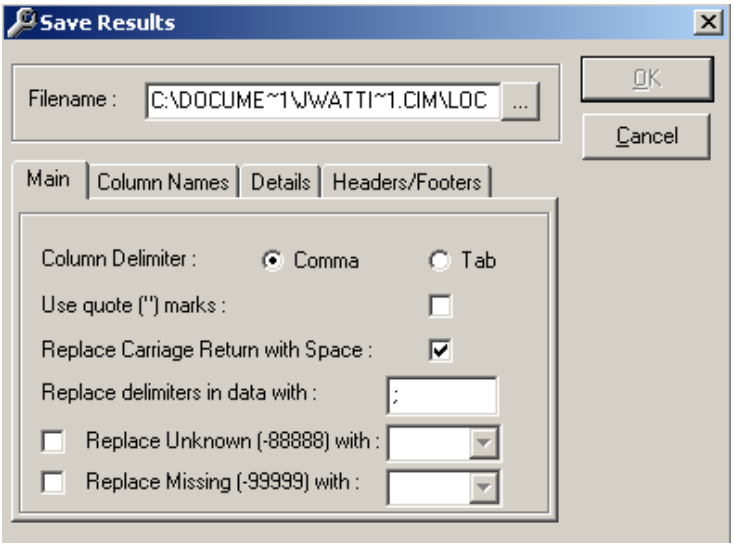
To export your query results to a file, select your query on the queries tab in **QDM**.
If you wish, run a preview of it.



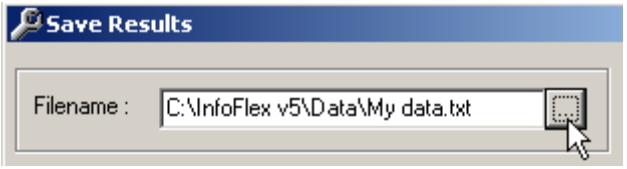
Go to the **File** menu and choose **Export** then **File** (or press the **Export to File** button).



InfoFlex displays the **Save Results** window.



In the **Filename** box press the **…** button. Navigate to a folder location and give the file a name.



On the tabs, make selections about the format that the file should be in.

Tab	Property	Description
Main	Column Delimiter	Should the file be column or tab delimited
	Use quote (“) marks	Tick if you wish double quotes to be used as a text qualifier. (ie double quotes placed at the start and end of each item of data)
	Replace carriage return with space	Tick if you wish to replace a carriage return in the data with a space (recommended)
	Replace delimiters in data with	If your chosen delimiter exists in the data, choose a character to replace it with
	Replace Unknown (-88888) with	If you wish to replace -88888 in the data, tick then type the replacement text.
	Replace Missing (-99999) with	If you wish to replace -99999 in the data, tick then type the replacement text.
Column Names	Include column names	Tick if you wish to include column headings in the file.
	Prefix	Enter any prefix to appear before the column headings (ie not attached to each column heading)
	Suffix	Enter any suffix to appear after the column headings (ie not attached to each column heading)
Details	Column delimiter	Should the file be column or tab delimited (same as the option on the **Main** tab)
	Code/meaning separator	Choose or type a separator for code and meaning in coded, MR and dictionary items.
	Character at start of data	If you wish, enter a character that should appear at the start of every piece of data. (Replaces **Use quote marks**)
	Character at end of data	If you wish, enter a character that should appear at the end of every piece of data. (Replaces **Use quote marks**)
	Enclose column headings with start/end characters	Tick if you wish the start and end characters to be applied to each column heading.
Headers/Footers	File Header	Enter header text which will be the first row of the file.
	File Footer	Enter footer text which will be the last row of the file.
	Row Prefix	Enter prefix text which will appear at the start of every row
	Row Suffix	Enter suffix text which will appear at the end of every row.

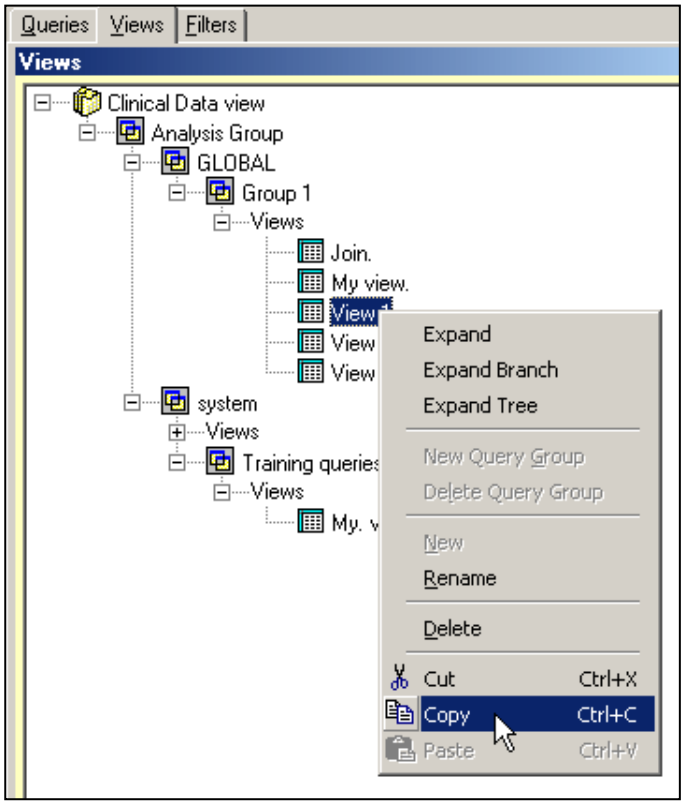
After making your selections, press OK to export the data. A confirmation message confirms that the data has been exported.

You can now open and review the exported data file.

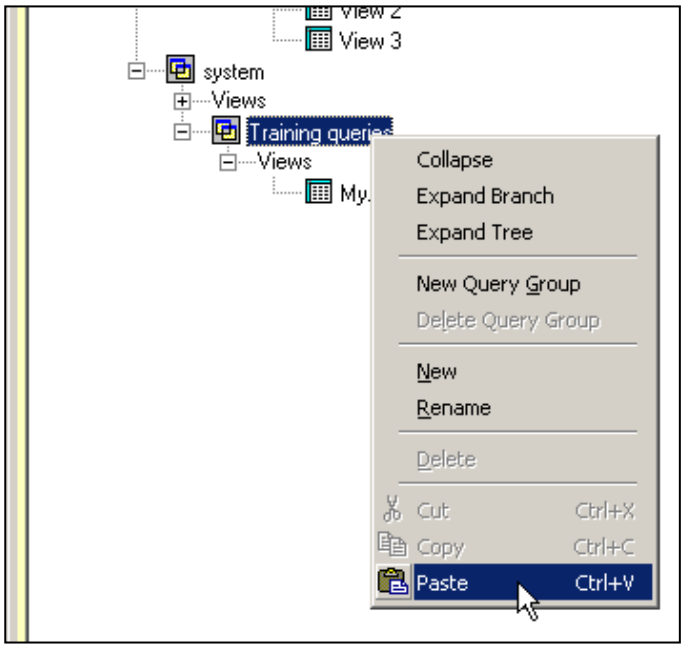
12 MOVING AND COPYING VIEWS, FILTERS AND QUERIES

Views, filters and queries can be cut, copied and pasted. You can use this facility to copy or move views, filters and queries between query groups. Cutting and pasting a query will move it from one group to another. Copying and pasting a query will make a copy of the query in a new group or in the same group.

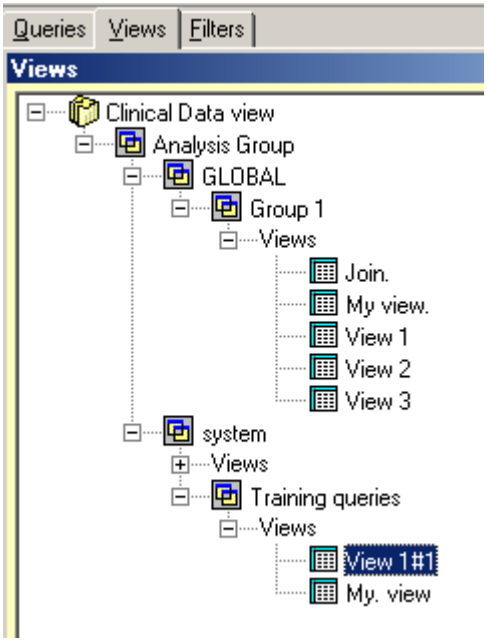
To cut or copy, right click the view, filter or query concerned and choose **Cut** or **Copy**.



To paste the query, right click the **group** name of the group you wish to copy or move it to and choose **Paste**.



The new view is displayed in the group to which you have moved or copied it.



Note that when views, filters or queries are copied, the name of the new view, filter or query has a number (eg "#1") appended to it so that the same name is not used twice.

13 SYNTAX DIFFERENCES BETWEEN SQL AND ACCESS

When functions are used in Views and Filters, the expressions are converted into SQL expressions and then SQL Server (or Access) evaluates the expressions, rather than InfoFlex. In some cases, views and filters need to be defined differently for Access. Some examples are listed below.

Note that if a database is moved between platforms, eg an SQL database is converted into an Access database, then any existing query syntax that is specific to the database platform will not be converted. Some queries might therefore fail because the syntax is not appropriate for the new platform.

13.1 Interval arguments in date functions

a) Access requires interval arguments to be in double quotes. These are inserted automatically by the formula builder when the database is Access.

   If the database is SQLServer and quotes are placed around the interval arguments, InfoFlex automatically removes the quotes.

b) SQLServer and Access use slightly different interval arguments. These are listed below.

Date Part	SQL Server	MS Access
Year	year, yy, yyyy	"yyyy"
Quarter	quarter, qq, q	"q"
Month	month, mm, m	"m"
Day of Year	dayofyear, dy, y	"y"
Day	day, dd, d	"d"
Week	week, wk, ww	"ww"
Day of Week	weekday, dw	"w"
Hour	hour, hh	"h"
Minute	minute, mi, n	"n"
Second	second, ss, s	"s"
Millisecond	millisecond, ms	-

For example:
In Access: DateDiff("d",*DateItem*,Now())
In SQL Server: DateDiff(day,*DateItem*, Now())
or DateDiff(dd,*DateItem*, Now())
or DateDiff(d,*DateItem*, Now())

13.2 Date calculations

When subtracting one date from another, SQLServer returns an error whereas Access returns a value. For this reason, functions (eg DATEDIFF) should always be used for calculations involving dates.

13.3 Functions in filters

The syntax for substituting null with a value is different in SQLServer and Access. This syntax would be used where a record has a null value in any of the items used in a filter calculation. Substituting the null value with another value such as zero to ensure that a record will be returned by the query.

In SQLServer, the syntax to use **0** instead of **Null** for a data item called **ValueItem** is as follows:
IsNULL(ValueItem,0)

In Access, the syntax to use **0** instead of **Null** for a data item called **ValueItem** is as follows:
(IIF(ValueItem,"",0))

See section 10.3 for full details.

13.4 Views and filters

13.4.1 IFNULL, IFMISSING, IFUNKNOWN, IFMISSINGORUNKNOWN

IFNULL, IFMISSING, IFUNKNOWN, IFMISSINGORUNKNOWN can be used in views but not in filters. In views these functions can be used both in SQLServer and in Access.

13.4.2 IsNull

SQLServer supports IsNull(arg1, arg2) in both views and filters, (although it is not listed in the formula builder). IsNull(arg1, arg2) cannot be used in Access.

Access supports IsNull(arg1) in both views and filters. IsNull(arg1) cannot be used in SQLServer.

13.4.3 IIF

IIF **cannot** be used in a filter in SQLServer.
IIF **can** be used in a filter in Access.

13.4.4 Count(Distinct)

Count(Distinct) cannot be used in a view in Access.

13.5 String concatenation

Strings in SQL Server should be concatenated using the **+** symbol whereas strings in Access should be concatenated using the **&** symbol. The arguments being concatenated should be strings rather than numbers.

Note that QDM allows you to enter the invalid syntax, however the query will fail to run.